

```
LLL          AAAAAAAAAA      TTTTTTTTTTTTTTTT
LLL          AAAAAAAAAA      TTTTTTTTTTTTTTTT
LLL          AAAAAAAAAA      TTTTTTTTTTTTTTTT
LLL          AAA            AAA    TTT
LLL          AAA            AAA    TTT
LLL          AAA            AAA    TTT
LLL          AAA            AAA    TTT
LLL          AAA            AAA    TTT
LLL          AAA            AAA    TTT
LLL          AAA            AAA    TTT
LLL          AAA            AAA    TTT
LLL          AAA            AAA    TTT
LLL          AAA            AAA    TTT
LLL          AAAAAAAAAAAAAAA   TTT
LLL          AAAAAAAAAAAAAAA   TTT
LLL          AAAAAAAAAAAAAAA   TTT
LLL          AAA            AAA    TTT
LLL          AAA            AAA    TTT
LLL          AAA            AAA    TTT
LLLLLLLLLLLLLLLLLLLLLL     AAA    TTT
LLLLLLLLLLLLLLLLLLLLLL     AAA    TTT
LLLLLLLLLLLLLLLLLLLLLL     AAA    TTT
```

```
LL      AAAAAA  TTTTTTTTTT  CCCCCCCC  PPPPPPPP
LL      AAAAAA  TTTTTTTTTT  CCCCCCCC  PPPPPPPP
LL      AA      AA      TT      CC      PP      PP
LL      AA      AA      TT      CC      PP      PP
LL      AA      AA      TT      CC      PP      PP
LL      AA      AA      TT      CC      PPPPPPPP
LL      AA      AA      TT      CC      PPPPPPPP
LL      AAAAAAAAAA  TT      CC      PP
LL      AAAAAAAAAA  TT      CC      PP
LL      AA      AA      TT      CC      PP
LL      AA      AA      TT      CC      PP
LLLLLLLLLLLL  AA      AA      TT      CCCCCCCC
LLLLLLLLLLLL  AA      AA      TT      CCCCCCCC
```

```
....
....
....
....
```

```
LL      IIIIII  SSSSSSSS
LL      IIIIII  SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LLLLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLLLL  IIIIII  SSSSSSSS
```



```
0001 0 %TITLE 'LAT Control Program'
0002 0 MODULE LATCP (
0003 0     LANGUAGE (BLISS32),
0004 0     MAIN = LCP MAIN,
0005 0     IDENT = 'V04-000'
0006 0 ) =
0007 1 BEGIN
0008 1
0009 1
0010 1 *****
0011 1 *
0012 1 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0013 1 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0014 1 *  ALL RIGHTS RESERVED.
0015 1 *
0016 1 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0017 1 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0018 1 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0019 1 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0020 1 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0021 1 *  TRANSFERRED.
0022 1 *
0023 1 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0024 1 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0025 1 *  CORPORATION.
0026 1 *
0027 1 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0028 1 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0029 1 *
0030 1 *
0031 1 *****
0032 1
0033 1
0034 1 ++
0035 1 FACILITY:    LAT Control Program
0036 1
0037 1 ABSTRACT:
0038 1
0039 1     This program controls and obtains counters and other information
0040 1     from the LAT terminal port driver.
0041 1
0042 1 ENVIRONMENT: VAX/VMS Operating System
0043 1
0044 1 AUTHOR:      Darrell Duffy , CREATION DATE: 2-July-1983
0045 1
0046 1 Previously modified by:
0047 1     Darrel Duffy
0048 1     Joe Marchesani
0049 1
0050 1 MODIFIED BY:
0051 1
0052 1     V03-009 RNG0009      Rod Gamache      16-Jul-1984
0053 1     Use system NODEname for /NODE=NAME qualifier if SYSSNODE
0054 1     does not translate.
0055 1     Allow for the Physical UCB pointer to be zero in the check
0056 1     for LT devices for shut requests.
0057 1
```

LATCP
V04-000

LAT Control Program

B 5
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMMASTER:[LAT.SRC]LATCP.B32;1

Page 2
(1)

58	0058	1	V03-008	RNG0008	Rod Gamache	7-Jun-1984
59	0059	1		Change IDENTIFICATION to SERVICE IDENTIFICATION.		
60	0060	1		With change in .CLD for /NAME and /IDENTIFICATION,		
61	0061	1		the LATCP program must default values on it's own.		
62	0062	1		Add HELP command.		
63	0063	1		Add START HISTORY command.		
64	0064	1		Add STOP HISTORY command.		
65	0065	1		Add /[NO]LOG qualifier to START, SET and STOP verbs.		
66	0066	1		Make proper determination of LAT terminals when shutting		
67	0067	1		down.		
68	0068	1				
69	0069	1	V03-007	RNG0007	Rod Gamache	8-May-1984
70	0070	1		Change HOST qualifier to NODE qualifier, as per documentation		
71	0071	1		change.		
72	0072	1		Change MULTICAST_RATE to MULTICAST_TIMER.		
73	0073	1				
74	0074	1	V03-006	RNG0006	Rod Gamache	1-May-1984
75	0075	1		Change LATCP to be a utility, instead of DCL command.		
76	0076	1				
77	0077	1	V03-005	LMP0221	L. Mark Pilant,	12-Apr-1984 14:42
78	0078	1		Change UCBSL_OWNUIC to ORBSL_OWNER and UCBSW_VPROT to		
79	0079	1		ORBSW_PROT.		
80	0080	1				
81	0081	1	V03-004	RNG0004	Rod Gamache	27-Mar-1984
82	0082	1		Add SHOW COUNTERS/DEVICE, SHOW COUNTERS/SERVERS and		
83	0083	1		SHOW SERVERS commands.		
84	0084	1		Add START/SET NODE_ID command.		
85	0085	1				
86	0086	1	V03-003	RNG0003	Rod Gamache	14-Mar-1984
87	0087	1		Fix references to NMADEF.		
88	0088	1				
89	0089	1	V03-002	RNG0002	Rod Gamache	2-Mar-1984
90	0090	1		Add circuit counters.		
91	0091	1				
92	0092	1	V03-001	RNG0001	Rod Gamache	31-Jan-1984
93	0093	1		Add new V5 message formats.		
94	0094	1				

LAT
V04

53

52

41

00

6E
20
20
21


```

: 96 0095 1 %SBTTL 'Definitions'
: 97 0096 1
: 98 0097 1
: 99 0098 1
: 100 0099 1
: 101 0100 1
: 102 0101 1 LIBRARY 'SYSS$LIBRARY:LIB';
: 103 0102 1 LIBRARY 'SHRLIB$:NMALIBRY';
: 104 0103 1 REQUIRE 'OBJ$:LATDEF';
: 105 0241 1
: 106 0242 1
: 107 0243 1
: 108 0244 1
: 109 0245 1
: 110 0246 1 LINKAGE
: 111 0247 1     ALONONPAGED = JSB (REGISTER = 1; REGISTER = 2),
: 112 0248 1     DEANONPAGED = JSB (REGISTER = 0)
: 113 0249 1         : NOPRESERVE (1,2,3),
: 114 0250 1     VERIFYCHAN = JSB (REGISTER = 0; REGISTER = 1)
: 115 0251 1         : NOPRESERVE (2,3,4,5),
: 116 0252 1     STARTSTOP = JSB ( REGISTER = 5 )
: 117 0253 1         : NOPRESERVE (2,3,4,5,6,7,8,9,10,11)
: 118 0254 1
: 119 0255 1
: 120 0256 1
: 121 0257 1 TABLE OF CONTENTS:
: 122 0258 1
: 123 0259 1
: 124 0260 1 FORWARD ROUTINE
: 125 0261 1     LCP_MAIN,
: 126 0262 1     LCP_QUAL,
: 127 0263 1     LCP_GETSTR,
: 128 0264 1     LCP_FIXNAME,
: 129 0265 1     LCPR_FINDDRIVER,
: 130 0266 1     LCP_START : NOVALUE,
: 131 0267 1     LCP_STOP : NOVALUE,
: 132 0268 1     LCP_SET,
: 133 0269 1     LCP_SHOW : NOVALUE,
: 134 0270 1     LCP_EXIT : NOVALUE,
: 135 0271 1     LCP_ZERO : NOVALUE,
: 136 0272 1     LCPK_SET,
: 137 0273 1     LCPK_START,
: 138 0274 1     LCPK_STOP,
: 139 0275 1     LCPK_ZERO : NOVALUE,
: 140 0276 1     LCPK_STARTHISTORY,
: 141 0277 1     LCPK_STOPHISTORY : NOVALUE,
: 142 0278 1     LCPK_IS LAT TERMINAL,
: 143 0279 1     LCP_SHOWCOUQUAL : NOVALUE,
: 144 0280 1     LCP_SHOWCOU : NOVALUE,
: 145 0281 1     LCP_SHOWNODECOU : NOVALUE,
: 146 0282 1     LCP_SHOWDEVCOU : NOVALUE,
: 147 0283 1     LCPR_SHOWDEVCOU,
: 148 0284 1     LCP_DISPDEVCOU,
: 149 0285 1     LCP_FINDCOU,
: 150 0286 1     LCP_SHOWSERVCOU : NOVALUE,
: 151 0287 1     LCP_SHOWHIST : NOVALUE,
: 152 0288 1     LCP_SHOWCHAR : NOVALUE,
:
: Main entry
: Convert all qualifiers
: Obtain a string qualifier
: Fixup a name string qualifier
: Is the port driver loaded?
: Start the driver
: Stop the driver
: Set the multicast message
: Process a SHOW command
: Exit LCP command
: Zero the counters
: Kernel mode set multicast routine
: Kernel mode start routine
: Kernel mode stop routine
: Kernel mode zero routine
: Kernel mode start history routine
: Kernel mode stop history routine
: Kernel mode find LAT terminal routine
: Show counters qualifiers
: Show counters
: Show node counters
: Show Ethernet device counters
: Show Ethernet device counters
: Display the Ethernet device ctrs
: Find the counter value
: Show LAT Server counters
: Show the history
: Show the characteristics
```



```

: 153      0289 1      LCP_SHOWSERVERS : NOVALUE,
: 154      0290 1      LCP_HELP : NOVALUE,
: 155      0291 1      LCP_CVTGROUPS : NOVALUE,
: 156      0292 1      LCP_FAOUTPUT : NOVALUE,
: 157      0293 1      LCPK_GETCOUNTERS,
: 158      0294 1      LCPK_GETSERVCTRS,
: 159      0295 1      LCPK_GETSERVERS,
: 160      0296 1      LCPK_GETHISTORY,
: 161      0297 1      LCPK_GETCOMMON
: 162      0298 1      ;
: 163      0299 1
: 164      0300 1
: 165      0301 1      MACROS:
: 166      0302 1
: 167      0303 1
: 168      0304 1      MACRO
: 169      0305 1
: 170      0306 1      Set up a static descriptor
: 171      0307 1
: 172      M 0308 1      DESCRIPTOR (STRING) =
: 173      0309 1          UPLIT (%CHARCOUNT (STRING), UPLIT BYTE (STRING))%,
: 174      0310 1
: 175      0311 1      Set up a dynamic descriptor
: 176      0312 1
: 177      M 0313 1      $SETDSC (DSC, SIZE, ADR) =
: 178      M 0314 1          (
: 179      M 0315 1              DSC [DSC$B_CLASS] = DSC [DSC$B_DTYPE] = 0;
: 180      M 0316 1              DSC [DSC$W_LENGTH] = SIZE;
: 181      M 0317 1              DSC [DSC$A_POINTER] = ADR
: 182      0318 1          )%,
: 183      0319 1
: 184      0320 1      Compare a pair of descriptors
: 185      0321 1
: 186      M 0322 1      $CMPDSC (DSC1, DSC2) =
: 187      M 0323 1          (
: 188      M 0324 1              CH$EQL
: 189      M 0325 1                  (
: 190      M 0326 1                      .DSC1 [DSC$W_LENGTH],
: 191      M 0327 1                      .DSC1 [DSC$A_POINTER],
: 192      M 0328 1                      MAX ( MIN ( .DSC1 [DSC$W_LENGTH], .DSC2 [DSC$W_LENGTH]), 3),
: 193      M 0329 1                      .DSC2 [DSC$A_POINTER],
: 194      M 0330 1                      ;
: 195      M 0331 1                  )
: 196      0332 1              )%,
: 197      0333 1
: 198      0334 1      LOG information if requested
: 199      0335 1
: 200      M 0336 1      LOG_SIGNAL [] =
: 201      M 0337 1          (
: 202      M 0338 1              IF (CLIS$PRESENT (QUAL_LOG) )
: 203      M 0339 1              THEN
: 204      M 0340 1                  SIGNAL (%REMAINING)
: 205      M 0341 1              ;
: 206      0342 1          )%,
: 207      0343 1
: 208      M 0344 1      LOG_OUTPUT [] =
: 209      M 0345 1          (
```

```

210 M 0346 1      IF (CLISPRESNT (QUAL_LOG) )
211 M 0347 1      THEN
212 M 0348 1      LIB$PUT_OUTPUT (%REMAINING)
213 M 0349 1      ;
214 M 0350 1      ;%
215 M 0351 1      ;
216 M 0352 1      ;
217 M 0353 1      ;
218 M 0354 1      ; Define FLAGS bits
219 M 0355 1      ;
220 M 0356 1      MACRO
221 M 0357 1      FLAG_V_SERVERS = 0,0,1,0%,
222 M 0358 1      FLAG_V_DEVICE  = 0,1,1,0%,
223 M 0359 1      FLAG_V_NODE    = 0,2,1,0%,
224 M 0360 1
225 M 0361 1      FLAG_L_ALL      = 0,0,32,0%
226 M 0362 1      ;
227 M 0363 1      ;
228 M 0364 1      ;
229 M 0365 1      ; EQUATED SYMBOLS:
230 M 0366 1      ;
231 M 0367 1      ;
232 M 0368 1      LITERAL
233 M 0369 1      TRUE = 1,
234 M 0370 1      FALSE = 0,
235 M 0371 1      SUCCESS = 1,
236 M 0372 1      FAILURE = 0,
237 M 0373 1
238 M 0374 1      LF = 10,          ; Line Feed character
239 M 0375 1      CR = 13,          ; Carriage return character
240 M 0376 1
241 M 0377 1      INPUT_LEN = 1024, ; Input command line length
242 M 0378 1      CMDLEN = 16,      ; Subcommand string length
243 M 0379 1      NUMLEN = 16,      ; Numbers
244 M 0380 1      ;
245 M 0381 1
246 M 0382 1      BIND
247 M 0383 1      LCP_PROMPT      = %ASCII 'LCP> ',
248 M 0384 1      QUAL_SUBCOMMAND = %ASCII 'SUBCOMMAND',
249 M 0385 1      QUAL_NAME       = %ASCII 'NODE NAME' : BLOCK [,BYTE],
250 M 0386 1      QUAL_ID        = %ASCII 'SERVICE IDENTIFICATION' : BLOCK [,BYTE],
251 M 0387 1      QUAL_GROUPS     = %ASCII 'GROUPS' : BLOCK [,BYTE],
252 M 0388 1      QUAL_RATING     = %ASCII 'RATING' : BLOCK [,BYTE],
253 M 0389 1      QUAL_SERV_RATING = %ASCII 'RATING SERVICE' : BLOCK [,BYTE],
254 M 0390 1      QUAL_SERV_NAME  = %ASCII 'SERVICE NAME' : BLOCK [,BYTE],
255 M 0391 1      QUAL_START      = %ASCII 'START' : BLOCK [,BYTE],
256 M 0392 1      QUAL_STOP       = %ASCII 'STOP' : BLOCK [,BYTE],
257 M 0393 1      QUAL_SHOW       = %ASCII 'SHOW' : BLOCK [,BYTE],
258 M 0394 1      QUAL_SET        = %ASCII 'SET' : BLOCK [,BYTE],
259 M 0395 1      QUAL_ZERO       = %ASCII 'ZERO' : BLOCK [,BYTE],
260 M 0396 1      QUAL_COUNTERS   = %ASCII 'COUNTERS' : BLOCK [,BYTE],
261 M 0397 1      QUAL_HISTORY    = %ASCII 'HISTORY' : BLOCK [,BYTE],
262 M 0398 1      QUAL_CHAR       = %ASCII 'CHARACTERISTICS' : BLOCK [,BYTE],
263 M 0399 1      QUAL_MCAST      = %ASCII 'MULTICAST TIMER' : BLOCK [,BYTE],
264 M 0400 1      QUAL_SERVERS    = %ASCII 'SERVERS' : BLOCK [,BYTE],
265 M 0401 1      QUAL_DEVICE     = %ASCII 'DEVICE' : BLOCK [,BYTE],
266 M 0402 1      QUAL_NODE       = %ASCII 'NODE' : BLOCK [,BYTE],
```



```

: 267      0403 1      QUAL_LOG          = %ASCID 'LOG' : BLOCK [,BYTE],
: 268      0404 1      QUAL_NODEID       = %ASCID 'NODE_IDENTIFICATION' : BLOCK [,BYTE],
: 269      0405 1      QUAL_NONE          = %ASCID 'none available' : BLOCK [,BYTE]
: 270      0406 1      ;
: 271      0407 1
: 272      0408 1      BIND
: 273      0409 1      ACTIVE = %ASCID 'active',
: 274      0410 1      INACTIVE = %ASCID 'inactive'
: 275      0411 1      ;
: 276      0412 1
: 277      0413 1
: 278      0414 1      BIND
: 279      L 0415 1      COUNTERSTR = %ASCID %STRING
: 280      L 0416 1      (
: 281      L 0417 1      '!',
: 282      L 0418 1      'LCP Node Counters!//',
: 283      L 0419 1      '10UL Receive frames!//',
: 284      L 0420 1      '10UL Receive errors!//',
: 285      L 0421 1      '10UL Receive duplicates!//',
: 286      L 0422 1      '10UL Transmit frames!//',
: 287      L 0423 1      '10UL Transmit errors!//',
: 288      L 0424 1      '10XL Last transmit failure code!//',
: 289      L 0425 1      '10UL Retransmissions!//',
: 290      L 0426 1      '10UL Circuit timeouts!//',
: 291      L 0427 1      '10UL Protocol errors!//',
: 292      L 0428 1      '10XL Protocol bit mask!//',
: 293      L 0429 1      '10UL Resource errors!//',
: 294      L 0430 1      '10UL No transmit buffer!//',
: 295      L 0431 1      '10UL Unit timeouts!//',
: 296      0432 1      );
: 297      0433 1
: 298      0434 1      BIND
: 299      L 0435 1      PROTOCOLMASK = %ASCID %STRING
: 300      L 0436 1      (
: 301      L 0437 1      %CHAR (CR, LF),
: 302      L 0438 1      ' Protocol errors include:',
: 303      L 0439 1      %CHAR (CR, LF)
: 304      0440 1      );
: 305      0441 1
: 306      0442 1
: 307      0443 1      BIND
: 308      L 0444 1      SERVCTRSTR = %ASCID %STRING
: 309      L 0445 1      (
: 310      L 0446 1      '!',
: 311      L 0447 1      'LCP Server Counters for !AS!//',
: 312      L 0448 1      '10UL Receive frames!//',
: 313      L 0449 1      '10UL Transmit frames!//',
: 314      L 0450 1      '10UL Retransmissions!//',
: 315      L 0451 1      '10UL Out of sequence frames!//',
: 316      L 0452 1      '10UL Invalid messages!//',
: 317      L 0453 1      '10UL Invalid slots!//',
: 318      L 0454 1      '!',
: 319      0455 1      );
: 320      0456 1
: 321      0457 1
: 322      0458 1      !
: 323      0459 1      ! NOTE: The following 2 tables must be kept in order!
```

```

324      0460 1 !      If the counters are to be displayed correctly.
325      0461 1      BIND
326      L 0462 1      DEVCTRSTR = %ASCID %STRING
327      L 0463 1      (
328      L 0464 1      '!',
329      L 0465 1      'LCP Ethernet Device Counters!/',
330      L 0466 1      '10UL Seconds since last zeroed!',
331      L 0467 1      '10UL Receive frames!',
332      L 0468 1      '10UL Receive errors!',
333      L 0469 1      '10UL Multicast frames received!',
334      L 0470 1      '10UL Receive bytes!',
335      L 0471 1      '10UL Multicast bytes received!',
336      L 0472 1      '10UL Transmit frames!',
337      L 0473 1      '10UL Transmit errors!',
338      L 0474 1      '10UL Multicast frames transmitted:',
339      L 0475 1      '10UL Transmit bytes!',
340      L 0476 1      '10UL Multicast bytes transmitted!',
341      L 0477 1      '10UL Frames sent, single collision!',
342      L 0478 1      '10UL Frames sent, multiple collisions!',
343      L 0479 1      '10UL Frames sent, initially deferred!',
344      L 0480 1      '10UL Transmit collision detect check failure!',
345      L 0481 1      '10UL Data overrun!',
346      L 0482 1      '10UL Local buffer errors!',
347      L 0483 1      '10UL System buffer unavailable!',
348      L 0484 1      '10UL User buffer unavailable!',
349      0485 1      );
350      0486 1
351      0487 1      BIND
352      0488 1      CTRNAMES = UPLIT
353      0489 1      (
354      0490 1      LONG (NMASC_CTLIN_ZER),
355      0491 1      LONG (NMASC_CTLIN_DBR),
356      0492 1      LONG (NMASC_CTLIN_RFL),
357      0493 1      LONG (NMASC_CTLIN_MBL),
358      0494 1      LONG (NMASC_CTLIN_BRC),
359      0495 1      LONG (NMASC_CTLIN_MBY),
360      0496 1      LONG (NMASC_CTLIN_DBS),
361      0497 1      LONG (NMASC_CTLIN_SFL),
362      0498 1      LONG (NMASC_CTLIN_MBS),
363      0499 1      LONG (NMASC_CTLIN_BSN),
364      0500 1      LONG (NMASC_CTLIN_MSN),
365      0501 1      LONG (NMASC_CTLIN_BS1),
366      0502 1      LONG (NMASC_CTLIN_BSM),
367      0503 1      LONG (NMASC_CTLIN_BID),
368      0504 1      LONG (NMASC_CTLIN_CDC),
369      0505 1      LONG (NMASC_CTLIN_OVR),
370      0506 1      LONG (NMASC_CTLIN_LBE),
371      0507 1      LONG (NMASC_CTLIN_SBU),
372      0508 1      LONG (NMASC_CTLIN_UBU),
373      0509 1      -1
374      0510 1      );
375      0511 1
376      0512 1      BIND
377      L 0513 1      RCVERRMASK = %ASCID %STRING
378      L 0514 1      (
379      L 0515 1      %CHAR (CR, LF),
380      L 0516 1      'Receive errors include:',

```



```

381 L 0517 1 %CHAR (CR, LF)
382 0518 1 );
383 0519 1
384 0520 1 BIND
385 L 0521 1 XMTERMASK = %ASCID %STRING
386 L 0522 1 (
387 L 0523 1 %CHAR (CR, LF),
388 L 0524 1 ' Transmit errors include:',
389 L 0525 1 %CHAR (CR, LF)
390 0526 1 );
391 0527 1
392 0528 1 BIND
393 L 0529 1 faostr1 = %ascid %string
394 L 0530 1 (
395 L 0531 1 '!/',
396 L 0532 1 Time Type Slots Idx Seq Idx Seq Seq Ack DATA...',
397 L 0533 1 '!/',
398 0534 1 ),
399 L 0535 1 faostr2 = %ascid %string
400 L 0536 1 (
401 L 0537 1 '!14<!%T!> !XB !XB !XB !XB !XB !XB !XB !XB !XB !XB!',
402 L 0538 1 '!XB !XB !XB !XB !XB !XB !XB !XB !XB !XB !XB !XB !/' ,
403 0539 1 );
404 0540 1
405 0541 1
406 0542 1 List of device names to try in order. A logical name allows us to
407 0543 1 define a new datalink device to use or to use a controller other than
408 0544 1 the first.
409 0545 1 We assume that the UNA is first, and we allow the QNA driver to be
410 0546 1 used also.
411 0547 1
412 0548 1 BIND
413 0549 1 DEVNAMES = UPLIT
414 0550 1 (
415 0551 1 %ASCID 'LATS$DEVICE',
416 0552 1 %ASCID 'XEA0:',
417 0553 1 %ASCID 'XQA0:',
418 0554 1 0
419 0555 1 );
420 0556 1
421 0557 1
422 0558 1 The list of parameters for the datalink driver to start up. We include
423 0559 1 the protocol type here as well as the other parameters to allow the device
424 0560 1 to be started with or without DECnet.
425 0561 1
426 0562 1
427 0563 1 BIND
428 0564 1 SETPARM = UPLIT
429 0565 1 (
430 0566 1 WORD (NMASC_PCLI_BUS), ! Buffer size
431 0567 1 LONG (1500),
432 0568 1 WORD (NMASC_PCLI_BFN), ! Number of buffers
433 0569 1 LONG (1),
434 0570 1 WORD (NMASC_PCLI_PAD), ! Pad short buffers (NO!)
435 0571 1 LONG (NMASC_STATE_OFF),
436 0572 1 WORD (NMASC_PCLI_PTY), ! Protocol type
437 0573 1 LONG (%X'0480'), ! Digital assigned protocol type

```

```
438 0574 1 WORD (NMASC_PCLI_PHA), ! Physical address
439 0575 1 WORD (2), ! Length of string
440 0576 1 WORD (NMASC_LINMC_SDF), ! Use default setting
441 0577 1 WORD (NMASC_PCLI_PRM), ! Promiscuous mode
442 0578 1 LONG (NMASC_STATE_OFF),
443 0579 1 WORD (NMASC_PCLI_DCH), ! Data chaining
444 0580 1 LONG (NMASC_STATE_OFF),
445 0581 1 WORD (NMASC_PCLI_CRC), ! Generate crc on transmit
446 0582 1 LONG (NMASC_STATE_ON),
447 0583 1 WORD (NMASC_PCLI_CON), ! Controller mode
448 0584 1 LONG (NMASC_LINCN_NOR), ! Normal
449 0585 1 )
450 0586 1 SETPARMEND = UPLIT BYTE(0),
451 0587 1 SETPARMDSC = UPLIT (SETPARMEND - SETPARM, SETPARM)
452 0588 1 ;
453 0589 1
454 0590 1 !
455 0591 1 ! Build a list of Global Host Node protocol error bit values and their names
456 0592 1 !
457 0593 1
458 M 0594 1 MACRO GHBERRBITS [BITNAME, MEANING] =
459 M 0595 1 $BITPOSITION ( %NAME ('GHBSV_',BITNAME) ),
460 M 0596 1 %ASCID MEANING
461 0597 1 %;
462 0598 1
463 0599 1 BIND
464 0600 1 PROTOCOLBITS =
465 0601 1 UPLIT
466 0602 1 (
467 P 0603 1 GHBERRBITS
468 P 0604 1 (
469 P 0605 1 START, ' Invalid start message received',
470 P 0606 1 CSBZERO, ' Zero node circuit index received',
471 P 0607 1 CSBRANGE, ' Node circuit index out of range',
472 P 0608 1 CSBINVALID, ' Node circuit sequence invalid',
473 P 0609 1 CSBSTALE, ' Node circuit index no longer valid',
474 P 0610 1 INVALIDSEQ, ' Invalid sequence number received in start message',
475 P 0611 1 HALT, ' Circuit was forced to halt',
476 P 0612 1 INVALIDREMID, ' Invalid server slot index',
477 P 0613 1 INVALIDLOCID, ' Invalid node slot index',
478 P 0614 1 BADCREBITS, ' Invalid credit field or too many credits used',
479 P 0615 1 REPCREATE, ' Repeat create of slot by server',
480 P 0616 1 REPDISC, ' Repeat disconnect of slot by server',
481 0617 1 ),
482 0618 1 0, 0
483 0619 1 );
484 0620 1 !
485 0621 1 !
486 0622 1 ! Define general error bit generation macro
487 0623 1 !
488 0624 1
489 M 0625 1 MACRO ERRBITS [BITVALUE, MEANING] =
490 M 0626 1 BITVALUE,
491 M 0627 1 %ASCID MEANING
492 0628 1 %;
493 0629 1 !
494 0630 1 !
```



```

: 495      0631 1  ! Build a list of Receive error bit values
: 496      0632 1  !
: 497      0633 1  BIND
: 498      0634 1      RCVERRBITS =
: 499      0635 1      UPLIT
: 500      0636 1      (
: 501      P 0637 1      ERRBITS
: 502      P 0638 1      (
: 503      P 0639 1          0, : CRC errors',
: 504      P 0640 1          1, : Framing errors',
: 505      P 0641 1          2, : Frame length error'
: 506      0642 1      ),
: 507      0643 1      0, 0
: 508      0644 1      );
: 509      0645 1
: 510      0646 1  !
: 511      0647 1  ! Build a list of Transmit error bit values
: 512      0648 1  !
: 513      0649 1  BIND
: 514      0650 1      XMERRBITS =
: 515      0651 1      UPLIT
: 516      0652 1      (
: 517      P 0653 1      ERRBITS
: 518      P 0654 1      (
: 519      P 0655 1          0, : More than 16 retry failures',
: 520      P 0656 1          1, : Loss of carrier',
: 521      P 0657 1          2, : Unknown',
: 522      P 0658 1          3, : Unknown',
: 523      P 0659 1          4, : Frame length error',
: 524      P 0660 1          5, : Late collision',
: 525      0661 1      ),
: 526      0662 1      0, 0
: 527      0663 1      );
: 528      0664 1
```

```
530 0665 1 |
531 0666 1 | OWN STORAGE:
532 0667 1 |
533 0668 1 |
534 0669 1 |
535 0670 1 | These are the strings and descriptors used to parse commands and qualifiers.
536 0671 1 |
537 0672 1 | OWN
538 0673 1 | INPUT_LINE : VECTOR [INPUT_LEN, BYTE],
539 0674 1 | INPUT_DSC : BLOCK [DSC$K_Z_BLN, BYTE],
540 0675 1 | CMDBUF_DSC : BLOCK [DSC$K_Z_BLN, BYTE],
541 0676 1 | SUBCMD : VECTOR [CMDLEN, BYTE],
542 0677 1 | SUBCMD_DSC : BLOCK [DSC$K_Z_BLN, BYTE],
543 0678 1 | NAME : VECTOR [GHB$K_NAMELEN, BYTE],
544 0679 1 | NAMEDSC : BLOCK [DSC$K_Z_BLN, BYTE],
545 0680 1 | SERV : VECTOR [GHB$K_NAMELEN, BYTE],
546 0681 1 | SERV_DSC : BLOCK [DSC$K_Z_BLN, BYTE],
547 0682 1 | ID : VECTOR [GHB$K_IDLEN, BYTE],
548 0683 1 | ID_DSC : BLOCK [DSC$K_Z_BLN, BYTE],
549 0684 1 | NODE_ID : VECTOR [GHB$K_IDLEN, BYTE],
550 0685 1 | NODE_ID_DSC : BLOCK [DSC$K_Z_BLN, BYTE],
551 0686 1 | NUMLST : VECTOR [NUMLEN, BYTE],
552 0687 1 | NUMLST_DSC : BLOCK [DSC$K_Z_BLN, BYTE],
553 0688 1 |
554 0689 1 | GROUPS : BLOCK [32, BYTE],
555 0690 1 | RATING : LONG,
556 0691 1 | SERV_RATING : LONG,
557 0692 1 | VERSION : LONG,
558 0693 1 | ECO : LONG,
559 0694 1 | MCASTIMR : LONG,
560 0695 1 |
561 0696 1 | FLAGS : BLOCK [1, LONG] ! Flags word
562 0697 1 | ;
563 0698 1 |
564 0699 1 | LITERAL OUTBUFSIZE = 1024,
565 0700 1 | GRPBUFSIZE = 1024;
566 0701 1 |
567 0702 1 | These data structures are those used in starting, stopping and showing
568 0703 1 | the counters, characteristics and history buffer of the driver.
569 0704 1 |
570 0705 1 |
571 0706 1 | OWN
572 0707 1 | LCP$L_AREA : REF BLOCK [, BYTE],
573 0708 1 | AREA : BLOCK [MAX (GHB$K_COMM_AREA, GHB$K_HISTSIZE,
574 0709 1 | (GHB$K_NAMELEN+LCB_C_LENGTH+1)*GHB$K_ACT_CSBS+GHB$K_INACT_CSBS)),
575 0710 1 | BYTE],
576 0711 1 | AREA_RET_SIZE : LONG,
577 0712 1 | OUTDSC : BLOCK [DSC$K_Z_BLN, BYTE],
578 0713 1 | OUTBUF : VECTOR [OUTBUFSIZE, BYTE],
579 0714 1 | GRPDSC : BLOCK [DSC$K_Z_BLN, BYTE],
580 0715 1 | GRPBUF : VECTOR [GRPBUFSIZE, BYTE],
581 0716 1 | ;
582 0717 1 |
583 0718 1 |
584 0719 1 |
585 0720 1 | EXTERNAL REFERENCES:
586 0721 1 |
```



```
: 587 0722 1
: 588 0723 1 EXTERNAL ROUTINE
: 589 0724 1 CLISGET VALUE : ADDRESSING MODE (GENERAL),
: 590 0725 1 CLISPRESENT : ADDRESSING MODE (GENERAL),
: 591 0726 1 OTSSCVT_TL : ADDRESSING MODE (GENERAL),
: 592 0727 1 OTSSCVT_L_TL : ADDRESSING MODE (GENERAL),
: 593 0728 1 STRSTRIM : ADDRESSING MODE (GENERAL),
: 594 0729 1 STR$UPCASE : ADDRESSING MODE (GENERAL),
: 595 0730 1 LIB$PUT_OUTPUT : ADDRESSING MODE (GENERAL),
: 596 0731 1 LIB$FFS : ADDRESSING MODE (GENERAL),
: 597 0732 1 LIB$SKPC : ADDRESSING MODE (GENERAL),
: 598 0733 1 LIB$GET_INPUT : ADDRESSING MODE (GENERAL),
: 599 0734 1 LIB$GET_FOREIGN : ADDRESSING MODE (GENERAL),
: 600 0735 1 CLISDCL_PARSE : ADDRESSING MODE (GENERAL),
: 601 0736 1 CLISDISPATCH : ADDRESSING MODE (GENERAL),
: 602 0737 1 IOCSVERIFYCHAN : VERIFYCHAN ADDRESSING MODE (GENERAL),
: 603 0738 1 EXESALONONPAGED : ALONONPAGED ADDRESSING MODE (GENERAL),
: 604 0739 1 EXESDEANONPAGED : DEANONPAGED ADDRESSING MODE (GENERAL)
: 605 0740 1 ;
: 606 0741 1
: 607 0742 1 EXTERNAL
: 608 0743 1 LCP_COMMANDS, : Command table
: 609 0744 1 IOCSGL_DPTLIST : LONG : HEAD OF LIST OF DPTS
: 610 0745 1 ADDRESSING_MODE (GENERAL)
: 611 0746 1 ;
: 612 0747 1
: 613 0748 1 EXTERNAL LITERAL
: 614 0749 1 CLIS_ABSENT,
: 615 0750 1 LCPS_IVCMD,
: 616 0751 1 LCPS_IVQUAL,
: 617 0752 1 LCPS_NOTLOADED,
: 618 0753 1 LCPS_NOTINITED,
: 619 0754 1 LCPS_SET,
: 620 0755 1 LCPS_NOTSET,
: 621 0756 1 LCPS_STARTED,
: 622 0757 1 LCPS_NOTSTARTED,
: 623 0758 1 LCPS_STOPPED,
: 624 0759 1 LCPS_NOTSTOPPED,
: 625 0760 1 LCPS_NOHISTORY,
: 626 0761 1 LCPS_NOSTARTHIST,
: 627 0762 1 LCPS_NOTFROMLAT,
: 628 0763 1 LCPS_NOSERVERS,
: 629 0764 1 LCPS_INTERNAL
: 630 0765 1 ;
: 631 0766 1
```

```

: 633 0767 1 %SBTTL 'LCP MAIN Main entry point'
: 634 0768 1 ROUTINE LCP_MAIN =
: 635 0769 1
: 636 0770 1 !++
: 637 0771 1 FUNCTIONAL DESCRIPTION:
: 638 0772 1
: 639 0773 1 Main entry point for LAT control program. Obtain the command
: 640 0774 1 and do the main dispatch.
: 641 0775 1
: 642 0776 1 FORMAL PARAMETERS:
: 643 0777 1
: 644 0778 1 NONE
: 645 0779 1
: 646 0780 1 IMPLICIT INPUTS:
: 647 0781 1
: 648 0782 1 Foreign command read
: 649 0783 1
: 650 0784 1 IMPLICIT OUTPUTS:
: 651 0785 1
: 652 0786 1 NONE
: 653 0787 1
: 654 0788 1 ROUTINE VALUE:
: 655 0789 1 COMPLETION CODES:
: 656 0790 1
: 657 0791 1 exit status or signalled errors
: 658 0792 1
: 659 0793 1 SIDE EFFECTS:
: 660 0794 1
: 661 0795 1 ldriver controlled or information obtained.
: 662 0796 1
: 663 0797 1 --
: 664 0798 1
: 665 0799 2 BEGIN
: 666 0800 2
: 667 0801 2 LOCAL
: 668 0802 2 STATUS
: 669 0803 2 ;
: 670 0804 2
: 671 0805 2
: 672 0806 2 ! Go to see if the driver is loaded. We will also fail early
: 673 0807 2 ! If we don't have CMKRNL priv.
: 674 0808 2
: 675 0809 2 IF NOT (STATUS = $CMKRNL (ROUTIN = LCPK_FINDDRIVER) )
: 676 0810 2 THEN
: 677 0811 2 RETURN .STATUS
: 678 0812 2
: 679 0813 2
: 680 0814 2 ! Check for any foreign command
: 681 0815 2
: 682 0816 2 $SETDSC (CMDBUF_DSC, INPUT_LEN, INPUT_LINE);
: 683 0817 2 INPUT_DSC[DSC$A_POINTER] = INPUT_LINE; ! Set address of buffer
: 684 0818 2 IF LIB$GET_FOREIGN(CMDBUF_DSC, 0, INPUT_DSC)
: 685 0819 2 AND .INPUT_DSC [DSC$W_LENGTH] NEQ 0
: 686 0820 2 THEN
: 687 0821 2 BEGIN
: 688 0822 2 FLAGS [FLAG_L_ALL] = 0; ! Zero all flags
: 689 0823 2 STATUS = CLISDCL_PARSE(INPUT_DSC, LCP_COMMANDS, LIB$GET_INPUT,
```



```
.TITLE  LATCP LAT Control Program
.IDENT  \V04-000\

.PSECT  $SPLITS,NOWRT,NOEXE,2

.ASCII  \LCP> \<0><0><0>
.LONG   17694725
.ADDRESS P.AAB
.ASCII  \SUBCOMMAND\<0><0>
.LONG   17694730
.ADDRESS P.AAD
.ASCII  \NODE_NAME\<0><0><0>
.LONG   17694729
.ADDRESS P.AAF
.ASCII  \SERVICE_IDENTIFICATION\<0><0>

.LONG   17694742
.ADDRESS P.AAH
.ASCII  \GROUPS\<0><0>
.LONG   17694726
.ADDRESS P.AAJ
.ASCII  \RATING\<0><0>
.LONG   17694726
.ADDRESS P.AAL
.ASCII  \RATING_SERVICE\<0><0>

.LONG   17694734
.ADDRESS P.AAN
.ASCII  \SERVICE_NAME\
.LONG   17694732
.ADDRESS P.AAP
.ASCII  \START\<0><0><0>
.LONG   17694725
.ADDRESS P.AAR
```

```
50 4F 54 53 000B4 P.AAT: .ASCII \STOP\
      010E0004 000B8 P.AAS: .LONG 17694724
      00000000 000BC .ADDRESS P.AAT
57 4F 48 53 000C0 P.AAV: .ASCII \SHOW\
      010E0004 000C4 P.AAU: .LONG 17694724
      00000000 000C8 .ADDRESS P.AAV
20 54 45 53 000CC P.AAX: .ASCII \SET \
      010E0004 000D0 P.AAW: .LONG 17694724
      00000000 000D4 .ADDRESS P.AAX
4F 52 45 5A 000D8 P.AAZ: .ASCII \ZERO\
      010E0004 000DC P.AAY: .LONG 17694724
      00000000 000E0 .ADDRESS P.AAZ
      53 52 45 54 4E 55 4F 43 000E4 P.ABB: .ASCII \COUNTERS\
      010E0008 000EC P.ABA: .LONG 17694728
      00000000 000F0 .ADDRESS P.ABB
      00 59 52 4F 54 53 49 48 000F4 P.ABD: .ASCII \HISTORY\<0>
      010E0007 000FC P.ABC: .LONG 17694727
      00000000 00100 .ADDRESS P.ABD
53 43 49 54 53 49 52 45 54 43 41 52 41 48 43 00104 P.ABF: .ASCII \CHARACTERISTICS\<0>
      00 00113
      010E000F 00114 P.ABE: .LONG 17694735
      00000000 00118 .ADDRESS P.ABF
52 45 4D 49 54 5F 54 53 41 43 49 54 4C 55 4D 0011C P.ABH: .ASCII \MULTICAST_TIMER\<0>
      00 0012B
      010E000F 0012C P.ABG: .LONG 17694735
      00000000 00130 .ADDRESS P.ABH
      00 53 52 45 56 52 45 53 00134 P.ABJ: .ASCII \SERVERS\<0>
      010E0007 0013C P.ABI: .LONG 17694727
      00000000 00140 .ADDRESS P.ABJ
      00 00 45 43 49 56 45 44 00144 P.ABL: .ASCII \DEVICE\<0><0>
      010E0006 0014C P.ABK: .LONG 17694726
      00000000 00150 .ADDRESS P.ABL
      45 44 4F 4E 00154 P.ABN: .ASCII \NODE\
      010E0004 00158 P.ABM: .LONG 17694724
      00000000 0015C .ADDRESS P.ABN
      00 47 4F 4C 00160 P.ABP: .ASCII \LOG\<0>
      010E0003 00164 P.ABO: .LONG 17694723
      00000000 00168 .ADDRESS P.ABP
41 43 49 46 49 54 4E 45 44 49 5F 45 44 4F 4E 0016C P.ABR: .ASCII \NODE_IDENTIFICATION\<0>
      00 4E 4F 49 54 0017B
      010E0013 00180 P.ABQ: .LONG 17694739
      00000000 00184 .ADDRESS P.ABR
      00 65 6C 62 61 6C 69 61 76 61 20 65 6E 6F 6E 00188 P.ABT: .ASCII \none available\<0><0>
      00 00197
      010E000E 00198 P.ABS: .LONG 17694734
      00000000 0019C .ADDRESS P.ABT
      00 00 65 76 69 74 63 61 001A0 P.ABV: .ASCII \active\<0><0>
      010E0006 001A8 P.ABU: .LONG 17694726
      00000000 001AC .ADDRESS P.ABV
      65 76 69 74 63 61 6E 69 001B0 P.ABX: .ASCII \inactive\
      010E0008 001B8 P.ABW: .LONG 17694728
      00000000 001BC .ADDRESS P.ABX
6E 75 6F 43 20 65 64 6F 4E 20 50 43 4C 2F 21 001C0 P.ABZ: .ASCII \!/LCP Node Counters!//!10UL Receive f\
20 20 4C 55 30 31 21 2F 21 2F 21 73 72 65 74 001CF
      66 20 65 76 69 65 63 65 52 20 001DE
20 20 20 4C 55 30 31 21 2F 21 73 65 6D 61 72 001E8
21 73 72 6F 72 72 65 20 65 76 69 65 63 65 52 001F7
      .ASCII \rames!//!10UL Receive errors!//!10UL R\
```


LATCP
V04-000LAT Control Program
LCP_MAIN Main entry pointC 6
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1Page 16
(4)

74	61	63	69	6C	52	20	20	20	4C	55	30	31	21	2F	00206	
61	72	54	20	20	70	75	64	20	65	76	69	65	63	65	00210	.ASCII \eceive duplicates!//10UL Transmit fram\
					20	4C	55	30	31	21	2F	21	73	65	0021F	
61	72	54	20	20	6D	61	72	66	20	74	69	6D	73	6E	0022E	
21	2F	21	73	72	20	4C	55	30	31	21	2F	21	73	65	00238	.ASCII \es!//10UL Transmit errors!//10XL Las\
					6F	72	72	65	20	74	69	6D	73	6E	00247	
6C	69	61	66	20	73	61	4C	20	20	20	4C	58	30	31	00256	
4C	55	30	31	21	74	69	6D	73	6E	61	72	74	20	74	00260	.ASCII \t transmit failure code!//10UL Retrans\
					2F	21	65	64	6F	63	20	65	72	75	0026F	
					73	6E	61	72	74	65	52	20	20	20	0027E	
4C	55	30	31	21	2F	21	73	6E	6F	69	73	73	69	6D	00288	.ASCII \missions!//10UL Circuit timeouts!//10U\
65	6D	69	74	20	55	30	31	21	2F	21	73	74	75	6F	00297	
					63	6F	74	6F	72	50	20	20	20	4C	002A6	
72	65	20	6C	6F	58	30	31	21	2F	21	73	72	20	4C	002B0	.ASCII \L Protocol errors!//10XL Protocol bi\
50	20	20	20	4C	69	62	20	6C	6F	63	6F	74	6F	72	002BF	
					31	21	2F	21	6B	73	61	6D	20	74	002CE	
20	20	4C	55	30	20	65	63	72	75	6F	73	65	52	20	002D8	.ASCII \t mask!//10UL Resource errors!//10UL \
72	6F	72	72	65	20	20	4C	55	30	31	21	2F	21	73	002E7	
					20	20	4C	55	30	31	21	2F	21	73	002F6	
75	62	20	74	69	6D	73	6E	61	72	74	20	6F	4E	20	00300	.ASCII \ No transmit buffer!//10UL Unit timeou\
55	20	20	20	4C	55	30	31	21	2F	21	72	65	66	66	0030F	
					75	6F	65	6D	69	74	20	74	69	6E	0031E	
											2F	21	73	74	00328	.ASCII \ts!/\
												010E016C			0032C	P.ABY: .LONG 17695084
												00000000			00330	.ADDRESS P.ABZ
72	72	65	20	6C	6F	63	6F	74	6F	72	50	20	0A	0D	00334	P.ACB: .ASCII <13><10>\ Protocol errors include:\<13>
		0D	3A	65	64	75	6C	63	6E	69	20	73	72	6F	00343	
											00	00	00	0A	00350	.ASCII <10><0><0><0>
												010E001D			00354	P.ACA: .LONG 17694749
												00000000			00358	.ADDRESS P.ACB
6F	43	20	72	65	76	72	65	53	20	50	43	4C	2F	21	0035C	P.ACD: .ASCII \!/LCP Server Counters for !AS!//10UL \
21	53	41	21	20	72	6F	66	20	73	72	65	74	6E	75	0036B	
					20	20	4C	55	30	31	21	2F	21	2F	0037A	
73	65	6D	61	72	66	20	65	76	69	65	63	65	52	20	00384	.ASCII \ Receive frames!//10UL Transmit frames\
73	6E	61	72	54	20	20	20	4C	55	30	31	21	2F	21	00393	
					73	65	6D	61	72	66	20	74	69	6D	003A2	
61	72	74	65	52	20	20	20	4C	55	30	31	21	2F	21	003AC	.ASCII \!/10UL Retransmissions!//10UL Out o\
30	31	21	2F	21	73	6E	6F	69	73	73	69	6D	73	6E	003BB	
					6F	20	74	75	4F	20	20	20	4C	55	003CA	
6D	61	72	66	20	65	63	6E	65	75	71	65	73	20	66	003D4	.ASCII \f sequence frames!//10UL Invalid messa\
76	6E	49	20	20	20	4C	55	30	31	21	2F	21	73	65	003E3	
					61	73	73	65	6D	20	64	69	6C	61	003F2	
6E	49	20	20	20	4C	55	30	31	21	2F	21	73	65	67	003FC	.ASCII \ges!//10UL Invalid slots!//\<0><0>
2F	21	2F	21	73	74	6F	6C	73	20	64	69	6C	61	76	0040B	
												00	00		0041A	
												010E00BE			0041C	P.ACC: .LONG 17694910
												00000000			00420	.ADDRESS P.ACD
20	74	65	6E	72	65	68	74	45	20	50	43	4C	2F	21	00424	P.ACF: .ASCII \!/LCP Ethernet Device Counters!//10UL \
73	72	65	74	6E	75	6F	43	20	65	63	69	76	65	44	00433	
					20	4C	55	30	31	21	2F	21	2F	21	00442	
65	63	6E	69	73	20	73	64	6E	6F	63	65	53	20	20	0044C	.ASCII \ Seconds since last zeroed!//10UL Rec\
21	2F	21	64	65	6F	72	65	7A	20	74	73	61	6C	20	0045B	
					63	65	52	20	20	20	4C	55	30	31	0046A	
31	21	2F	21	73	65	6D	61	72	66	20	65	76	69	65	00474	.ASCII \eive frames!//10UL Receive errors!//10\
65	20	65	76	69	65	63	65	52	20	20	20	4C	55	30	00483	
					30	31	21	2F	21	73	72	6F	72	72	00492	
20	74	73	61	63	69	74	6C	75	4D	20	20	20	4C	55	0049C	.ASCII \UL Multicast frames received!//10UL \

LATCP
V04-000

LAT Control Program
LCP_MAIN Main entry point

D 6
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 17
(4)

```
64 65 76 69 65 63 65 72 20 73 65 6D 61 72 66 004AB
2F 21 73 65 74 79 62 20 65 76 69 65 63 65 52 004BA
61 63 69 74 6C 75 4D 20 20 20 4C 55 30 31 21 004C4
72 20 73 65 74 79 62 20 74 73 004D3
20 4C 55 30 31 21 2F 21 64 65 76 69 65 63 65 004E2
6D 61 72 66 20 74 69 6D 73 6E 61 72 54 20 20 004EC
20 4C 55 30 31 21 2F 21 73 65 004FB
6F 72 72 65 20 74 69 6D 73 6E 61 72 54 20 20 0050A
6C 75 4D 20 20 20 4C 55 30 31 21 2F 21 73 72 00514
61 72 73 66 20 74 73 61 72 54 20 20 00523
64 65 74 74 69 6D 73 6E 61 72 74 20 73 69 74 00532
73 6E 61 72 54 20 20 20 61 72 74 20 73 65 6D 0053C
21 73 65 74 79 62 20 74 69 6D 0054B
63 69 74 6C 75 4D 20 20 20 4C 55 30 31 21 2F 0055A
73 6E 61 72 74 20 73 65 74 79 62 20 74 73 61 00564
31 21 2F 21 64 65 74 74 69 6D 00573
65 73 20 73 65 6D 61 72 46 20 20 20 4C 55 30 00582
6C 6C 6F 63 20 65 6C 67 6E 69 73 20 2C 74 6E 0058C
30 31 21 2F 21 6E 6F 69 73 69 0059B
6E 65 73 20 73 65 6D 61 72 46 20 20 20 4C 55 005AA
6C 6F 63 20 65 6C 70 69 74 6C 75 6D 20 2C 74 005B4
21 2F 21 73 6E 6F 69 73 69 6C 005C3
73 20 73 65 6D 61 72 46 20 20 20 4C 55 30 31 005D2
20 79 6C 6C 61 69 74 69 6E 69 20 2C 74 6E 65 005DC
69 6D 73 6E 61 72 54 20 20 20 4C 55 30 31 21 005EB
74 65 64 20 6E 6F 69 73 69 6C 6C 6F 69 74 63 65 005FA
20 4C 55 30 31 21 2F 21 65 72 75 6C 69 61 66 00604
21 6E 75 72 72 65 76 6F 20 61 74 61 44 20 20 00613
4C 20 20 20 4C 55 30 31 21 2F 21 73 72 6F 00622
72 72 65 20 72 65 66 66 75 62 20 6C 61 63 6F 0062C
79 53 20 20 20 4C 55 30 31 21 2F 21 73 72 6F 0063B
65 66 66 75 62 20 6D 65 74 73 0064A
2F 21 65 6C 62 61 76 61 6E 75 20 72 00654
75 62 20 72 65 73 55 20 20 4C 55 30 31 21 00663
61 76 61 6E 75 20 72 65 66 69 00672
73 55 20 20 20 4C 55 30 31 21 0067C
61 76 61 6E 75 20 72 65 66 69 0068B
73 55 20 20 20 4C 55 30 31 21 0069A
61 76 61 6E 75 20 72 65 66 69 006A4
2F 21 65 6C 62 61 76 61 6E 75 20 72 65 66 69 006AC
010E0288 006AC P.ACE: 006AC 17695368
00000000 006B0 P.ACF: 006B0 0
00000000 006B4 P.ACG: 006B4 0
00003F2 006B8 006B8 1010
0000426 006BC 006BC 1062
00003F4 006C0 006C0 1012
00003E8 006C4 006C4 1000
00003EA 006C8 006C8 1002
00003F3 006CC 006CC 1011
0000424 006D0 006D0 1060
0000A8D 006D4 006D4 2701
00003E9 006D8 006D8 1001
0000A8E 006DC 006DC 2702
00003F6 006E0 006E0 1014
00003F7 006E4 006E4 1015
00003F5 006E8 006E8 1013
0000425 006EC 006EC 1061
0000428 006F0 006F0 1064
```

LAT
V04


```
00000411 006F4 .LONG 1041
00000429 006F8 .LONG 1065
0000042A 006FC .LONG 1066
FFFFFFF 00700 .LONG -1
6F 72 72 65 20 65 76 69 65 63 65 52 20 0A 0D 00704 P.ACI: .ASCII <13><10>\ Receive errors include:\<13>
0D 3A 65 64 75 6C 63 6E 69 20 73 72 00713
0A 0071F .ASCII <10>
010E001C 00720 P.ACH: .LONG 17694748
00000000' 00724 .ADDRESS P.ACI
72 72 65 20 74 69 6D 73 6E 61 72 54 20 0A 0D 00728 P.ACK: .ASCII <13><10>\ Transmit errors include:\<13>
0D 3A 65 64 75 6C 63 6E 69 20 73 72 6F 00737
00 00 0A 00744 .ASCII <10><0><0><0>
010E001D 00748 P.ACJ: .LONG 17694749
00000000' 0074C .ADDRESS P.ACK
20 20 20 20 65 6D 69 54 20 20 20 20 20 2F 21 00750 P.ACM: .ASCII \!/ Time Type Slots Idx Seq I\
49 20 73 74 6F 6C 53 20 65 70 79 54 20 20 20 0075F
63 41 20 71 65 53 20 20 20 71 65 53 20 78 64 0076E
00 2F 21 2E 2E 2E 41 54 41 44 20 20 20 78 64 00778
20 20 6B 00787 .ASCII \dx Seq Seq Ack DATA...!\<0><0>
00 00796
00 00797 .ASCII <0>
010E0045 00798 P.ACL: .LONG 17694789
00000000' 0079C .ADDRESS P.ACM
42 58 21 20 20 3E 21 54 25 21 3C 34 31 21 20 007A0 P.ACO: .ASCII \ !14<!\%T!> !XB !XB !XB !XB !XB\
20 20 42 58 21 20 20 20 20 20 20 20 20 20 20 007AF
58 21 20 42 58 21 20 20 20 20 20 20 20 20 20 007BE
20 20 42 58 21 20 20 20 20 20 20 20 20 20 20 007C8
20 20 42 58 21 20 20 20 20 20 20 20 20 20 20 007D7
42 58 21 20 42 58 21 20 20 42 58 21 20 42 58 007E6
20 42 58 21 20 20 42 58 21 20 42 58 21 20 20 007F0
00 2F 21 20 42 58 21 20 20 42 58 21 20 20 20 007FF
20 42 58 21 0080E
00 00818 .ASCII \XB !XB !XB !/\<0><0>
00 00827
010E0086 00828 P.ACN: .LONG 17694854
00000000' 0082C .ADDRESS P.ACO
00 00 45 43 49 56 45 44 24 54 41 4C 00830 P.ACR: .ASCII \LAT$DEVICE\<0><0>
010E000A 0083C P.ACQ: .LONG 17694730
00000000' 00840 .ADDRESS P.ACR
00 00 00 3A 30 41 45 58 00844 P.ACT: .ASCII \XEA0:\<0><0><0>
010E0005 0084C P.ACS: .LONG 17694725
00000000' 00850 .ADDRESS P.ACT
00 00 00 3A 30 41 51 58 00854 P.ACV: .ASCII \XQA0:\<0><0><0>
010E0005 0085C P.ACU: .LONG 17694725
00000000' 00860 .ADDRESS P.ACV
00000000' 00864 P.ACP: .ADDRESS P.ACQ, P.ACS, P.ACU
00000000' 00870 .LONG 0
0AF1 00874 P.ACW: .WORD 2801
000005DC 00876 .LONG 1500
0451 0087A .WORD 1105
00000001 0087C .LONG 1
0B1A 00880 .WORD 2842
00000001 00882 .LONG 1
0B0E 00886 .WORD 2830
00000460 00888 .LONG 1120
0B04 0088C .WORD 2820
0002 0088E .WORD 2
```

```
0004 00890 .WORD 4
0B18 00892 .WORD 2840
00000001 00894 .LONG 1
0B1B 00898 .WORD 2843
00000001 0089A .LONG 1
0B1C 0089E .WORD 2844
00000000 008A0 .LONG 0
0456 008A4 .WORD 1110
00000000 008A6 .LONG 0
00 008AA P.ACX: .BYTE 0
00000036 008AB .BLKB 1
00000000 008AC P.ACY: .LONG 54
00000000 008B0 .ADDRESS SETPARM
61 74 73 20 64 69 6C 61 76 6E 49 20 20 20 20 008B4 P.ADB: .ASCII \ Invalid start message received\<0>
65 63 65 72 20 65 67 61 73 73 65 6D 20 20 74 72 008C3
00 64 65 76 69 008D2
00 008D7 .ASCII <0>
010E0022 008D8 P.ADA: .LONG 17694754
00000000 008DC .ADDRESS P.ADB
63 20 65 64 6F 6E 20 6F 72 65 5A 20 20 20 20 008E0 P.ADD: .ASCII \ Zero node circuit index received\
65 72 20 78 65 64 6E 69 20 74 69 75 63 72 69 008EF
64 65 76 69 65 63 008FE
010E0024 00904 P.ADC: .LONG 17694756
00000000 00908 .ADDRESS P.ADD
69 75 63 72 69 63 20 65 64 6F 4E 20 20 20 20 0090C P.ADF: .ASCII \ Node circuit index out of range\<0>
20 66 6F 20 74 75 6F 20 78 65 64 6E 69 20 74 0091B
00 65 67 6E 61 72 0092A
010E0023 00930 P.ADE: .LONG 17694755
00000000 00934 .ADDRESS P.ADF
69 75 63 72 69 63 20 65 64 6F 4E 20 20 20 20 00938 P.ADH: .ASCII \ Node circuit sequence invalid\<0><0>
61 76 6E 69 20 65 63 6E 65 75 71 65 73 20 74 00947
00 00 64 69 6C 00956
00 0095B .ASCII <0>
010E0021 0095C P.ADG: .LONG 17694753
00000000 00960 .ADDRESS P.ADH
69 75 63 72 69 63 20 65 64 6F 4E 20 20 20 20 00964 P.ADJ: .ASCII \ Node circuit index no longer valid-
67 6E 6F 6C 20 6F 6E 20 78 65 64 6E 69 20 74 00973
00 64 69 6C 61 76 20 72 65 00982
00 0098B .ASCII <0>
010E0026 0098C P.ADI: .LONG 17694758
00000000 00990 .ADDRESS P.ADJ
71 65 73 20 64 69 6C 61 76 6E 49 20 20 20 20 00994 P.ADL: .ASCII \ Invalid sequence number received in \
65 72 20 72 65 62 6D 75 6E 20 65 63 6E 65 75 009A3
00 00 65 67 61 73 73 65 6D 20 74 72 61 74 73 009B2
00 009BC .ASCII \start message\<0><0><0>
00 009CB
010E0035 009CC P.ADK: .LONG 17694773
00000000 009D0 .ADDRESS P.ADL
73 61 77 20 74 69 75 63 72 69 43 20 20 20 20 009D4 P.ADN: .ASCII \ Circuit was forced to halt\<0><0>
74 6C 61 68 20 6F 74 20 64 65 63 72 6F 66 20 009E3
00 00 009F2
010E001E 009F4 P.ADM: .LONG 17694750
00000000 009F8 .ADDRESS P.ADN
72 65 73 20 64 69 6C 61 76 6E 49 20 20 20 20 009FC P.ADP: .ASCII \ Invalid server slot index\<0><0>
00 78 65 64 6E 69 20 74 6F 6C 73 20 72 65 76 00A0B
00 00A1A
00 00A1B .ASCII <0>
```


64	6F	6E	20	64	69	6C	61	76	6E	49	20	20	20	20	010E001D	00A1C	P.ADO:	.LONG	17694749	
		00	78	65	64	6E	69	20	74	6F	6C	73	20	65	00000000	00A20		.ADDRESS	P.ADP	
															010E001B	00A24	P.ADR:	.ASCII	\	Invalid node slot index\<0>
															00000000	00A33				
65	72	63	20	64	69	6C	61	76	6E	49	20	20	20	20	010E001B	00A40	P.ADQ:	.LONG	17694747	
6F	74	20	72	6F	20	64	6C	65	69	66	20	74	69	64	00000000	00A44		.ADDRESS	P.ADR	
															00000000	00A48	P.ADT:	.ASCII	\	Invalid credit field or too many cre\
															00000000	00A57				
															00000000	00A66				
															00000000	00A70		.ASCII	\	bits used\<0>\<0>\<0>
															00000000	00A7C	P.ADS:	.LONG	17694769	
61	65	72	63	20	74	61	65	70	65	52	20	20	20	20	010E0031	00A80		.ADDRESS	P.ADT	
73	20	79	62	20	74	6F	6C	73	20	66	6F	20	65	74	00000000	00A84	P.ADV:	.ASCII	\	Repeat create of slot by server\<0>
															00000000	00A93				
															00000000	00AA2				
															010E0023	00AA8	P.ADU:	.LONG	17694755	
															00000000	00AAC		.ADDRESS	P.ADV	
63	73	69	64	20	74	61	65	70	65	52	20	20	20	20	010E0023	00AB0	P.ADX:	.ASCII	\	Repeat disconnect of slot by server-
20	74	6F	6C	73	20	66	6F	20	74	63	65	6E	6E	6F	00000000	00ABF			\<0>	
															00000000	00ACE				
															010E0027	00AD8	P.ADW:	.LONG	17694759	
															00000000	00ADC		.ADDRESS	P.ADX	
															00000000	00AE0	P.ACZ:	.LONG	0	
															00000000	00AE4		.ADDRESS	P.ADA	
															00000001	00AEB		.LONG	1	
															00000000	00AEC		.ADDRESS	P.ADC	
															00000002	00AF0		.LONG	2	
															00000000	00AF4		.ADDRESS	P.ADE	
															00000003	00AF8		.LONG	3	
															00000000	00AFC		.ADDRESS	P.ADG	
															00000004	00B00		.LONG	4	
															00000000	00B04		.ADDRESS	P.ADI	
															0000000A	00B08		.LONG	10	
															00000000	00B0C		.ADDRESS	P.ADK	
															00000005	00B10		.LONG	5	
															00000000	00B14		.ADDRESS	P.ADM	
															00000006	00B18		.LONG	6	
															00000000	00B1C		.ADDRESS	P.ADO	
															00000007	00B20		.LONG	7	
															00000000	00B24		.ADDRESS	P.ADQ	
															00000008	00B28		.LONG	8	
															00000000	00B2C		.ADDRESS	P.ADS	
															00000009	00B30		.LONG	9	
															00000000	00B34		.ADDRESS	P.ADU	
															0000000B	00B38		.LONG	11	
															00000000	00B3C		.ADDRESS	P.ADW	
00	73	72	6F	72	72	65	20	43	52	43	20	20	20	20	00000000	00B40		.LONG	0, 0	
															00000000	00B48	P.AEA:	.ASCII	\	CRC errors\<0>\<0>
															00000000	00B57				
															010E000E	00B58	P.ADZ:	.LONG	17694734	
72	72	65	20	67	6E	69	6D	61	72	46	20	20	20	20	00000000	00B5C		.ADDRESS	P.AEA	
															00000000	00B60	P.AEC:	.ASCII	\	Framing errors\<0>\<0>
															00000000	00B6F				
															010E0012	00B74	P.AEB:	.LONG	17694738	
74	67	6E	65	6C	20	65	6D	61	72	46	20	20	20	20	00000000	00B78		.ADDRESS	P.AEC	
															00000000	00B7C	P.AEE:	.ASCII	\	Frame length error\<0>\<0>
															00000000	00B8B				

```

010E0016 00B94 P.AED: .LONG 17694742
00000000 00B98 .ADDRESS P.AEE
00000000 00B9C P.ADY: .LONG 0
00000000 00BA0 .ADDRESS P.ADZ
00000001 00BA4 .LONG 1
00000000 00BA8 .ADDRESS P.AEB
00000002 00BAC .LONG 2
00000000 00BB0 .ADDRESS P.AED
00000000 00BB4 .LONG 0, 0
31 20 6E 61 68 74 20 65 72 6F 4D 20 20 20 00BBC P.AEH: .ASCII \ More than 16 retry failures\<0>
65 72 75 6C 69 61 66 20 79 72 74 65 72 20 36 00BCB
00 73 00BDA
010E001F 00BDC P.AEG: .LONG 17694751
00000000 00BE0 .ADDRESS P.AEH
72 61 63 20 66 6F 20 73 73 6F 4C 20 20 20 00BE4 P.AEJ: .ASCII \ Loss of carrier\<0>
00 72 65 69 72 00BF3
010E0013 00BF8 P.AEI: .LONG 17694739
00000000 00BFC .ADDRESS P.AEJ
00 6E 77 6F 6E 6B 6E 55 20 20 20 20 00C00 P.AEL: .ASCII \ Unknown\<0>
010E000B 00C0C P.AEK: .LONG 17694731
00000000 00C10 .ADDRESS P.AEL
00 6E 77 6F 6E 6B 6E 55 20 20 20 20 00C14 P.AEN: .ASCII \ Unknown\<0>
010E000B 00C20 P.AEM: .LONG 17694731
00000000 00C24 .ADDRESS P.AEN
74 67 6E 65 6C 20 65 6D 61 72 46 20 20 20 00C28 P.AEP: .ASCII \ Frame length error\<0>\<0>
00 00 72 6F 72 72 65 20 68 00C37
010E0016 00C40 P.AEO: .LONG 17694742
00000000 00C44 .ADDRESS P.AEP
73 69 6C 6C 6F 63 20 65 74 61 4C 20 20 20 00C48 P.AER: .ASCII \ Late collision\<0>\<0>
00 00 6E 6F 69 00C57
010E0012 00C5C P.AEQ: .LONG 17694738
00000000 00C60 .ADDRESS P.AER
00000000 00C64 P.AEF: .LONG 0
00000000 00C68 .ADDRESS P.AEG
00000001 00C6C .LONG 1
00000000 00C70 .ADDRESS P.AEI
00000002 00C74 .LONG 2
00000000 00C78 .ADDRESS P.AEK
00000003 00C7C .LONG 3
00000000 00C80 .ADDRESS P.AEM
00000004 00C84 .LONG 4
00000000 00C88 .ADDRESS P.AEO
00000005 00C8C .LONG 5
00000000 00C90 .ADDRESS P.AEQ
00000000 00C94 .LONG 0, 0
00000000 00000000
```

.PSECT \$OWNS\$,NOEXE,2

```

00000 INPUT_LINE:
          .BLKB 1024
00400 INPUT_DSC:
          .BLKB 8
00408 CMDBUF_DSC:
          .BLKB 8
00410 SUBCMD: .BLKB 16
00420 SUBCMD_DSC:
          .BLKB 8
```


00428 NAME: .BLKB 16
00438 NAMEDSC: .BLKB 8
00440 SERV: .BLKB 16
00450 SERVDSC: .BLKB 8
00458 ID: .BLKB 64
00498 IDDSC: .BLKB 8
004A0 NODE_ID: .BLKB 64
004E0 NODE_IDDSC: .BLKB 8
004E8 NUMLST: .BLKB 16
004F8 NUMLSTDSC: .BLKB 8
00500 GROUPS: .BLKB 32
00520 RATING: .BLKB 4
00524 SERV_RATING: .BLKB 4
00528 VERSION: .BLKB 4
0052C ECO: .BLKB 4
00530 MCASTIMR: .BLKB 4
00534 FLAGS: .BLKB 4
00538 LCP\$SL_AREA: .BLKB 4
0053C AREA: .BLKB 2560
00F3C AREA_RET_SIZE: .BLKB 4
00F40 OUTDSC: .BLKB 8
00F48 OUTBUF: .BLKB 1024
01348 GRPDSC: .BLKB 8
01350 GRPBUF: .BLKB 1024

LCP_PROMPT= P.AAA
QUAL_SUBCOMMAND= P.AAC
QUAL_NAME= P.AAE
QUAL_ID= P.AAG
QUAL_GROUPS= P.AAI
QUAL_RATING= P.AAK
QUAL_SERV_RATING= P.AAM
QUAL_SERV_NAME= P.AAO
QUAL_START= P.AAQ
QUAL_STOP= P.AAS
QUAL_SHOW= P.AAU
QUAL_SET= P.AAW
QUAL_ZERO= P.AAY
QUAL_COUNTERS= P.ABA
QUAL_HISTORY= P.ABC
QUAL_CHAR= P.ABE
QUAL_MCAST= P.ABG
QUAL_SERVERS= P.ABI
QUAL_DEVICE= P.ABK
QUAL_NODE= P.ABM
QUAL_LOG= P.ABO
QUAL_NODEID= P.ABQ
QUAL_NONE= P.ABS
ACTIVE= P.ABU
INACTIVE= P.ABW
COUNTERSCTRSTR= P.ABY

PROTOCOLMASK= P.ACA
SERVCTRSTR= P.ACC
DEVCTRSTR= P.ACE
CTRNames= P.ACG
RCVERRMASK= P.ACH
XMTERRMASK= P.ACJ
FAOSTR1= P.ACL
FAOSTR2= P.ACN
DEVNames= P.ACP
SETPARM= P.ACW
SETPARMEND= P.ACX
SETPARMDS= P.ACY
PROTOCOLBITS= P.ACZ
RCVERRBITS= P.ADY
XMTERRBITS= P.AEF

.EXTRN CLISGET_VALUE, CLISPRESENT
.EXTRN OTSSCVT_TL, OTSSCVT_L_TL
.EXTRN STRSTRIM, STRSUPCASE
.EXTRN LIB\$PUT_OUTPUT, LIB\$FFS
.EXTRN LIB\$SKPC, LIB\$GET_INPUT
.EXTRN LIB\$GET_FOREIGN
.EXTRN CLISDCL_PARSE, CLISDISPATCH
.EXTRN IOCSVERIFYCHAN, EXESALONONPAGED
.EXTRN EXESDEANONPAGED
.EXTRN LCP_COMMANDS, IOCSGL_DPTLIST
.EXTRN CLIS_ABSENT, LCPS_IVCMD
.EXTRN LCPS_IVQUAL, LCPS_NOTLOADED
.EXTRN LCPS_NOTINITED, LCPS_SET
.EXTRN LCPS_NOTSET, LCPS_STARTED
.EXTRN LCPS_NOTSTARTED
.EXTRN LCPS_STOPPED, LCPS_NOTSTOPPED
.EXTRN LCPS_NOHISTORY, LCPS_NOSTARTHIST
.EXTRN LCPS_NOTFROMLAT
.EXTRN LCPS_NOSERVERS, LCPS_INTERNAL
.EXTRN SYSSCMKRNL

.PSECT \$CODE\$,NOWRT,2

007C 00000 LCP_MAIN:

56	00000000G	00	9E	00002	.WORD	Save R2,R3,R4,R5,R6	: 0768
55	00000000G	00	9E	00009	MOVAB	CLISDISPATCH, R6	:
54	00000000G	00	9E	00010	MOVAB	CLISDCL_PARSE, R5	:
53	0000'	CF	9E	00017	MOVAB	LIB\$GET_INPUT, R4	:
		7E	D4	0001C	MOVAB	INPUT_DSC, R3	:
		CF	9F	0001E	CLRL	-(SP)	: 0809
00000000G	00	02	FB	00022	PUSHAB	LCPK_FINDDRIVER	:
	52	50	D0	00029	CALLS	#2, SYSSCMKRNL	:
	04	52	E8	0002C	MOVL	R0, STATUS	:
	50	52	D0	0002F	BLBS	STATUS, 1\$	: 0811
			04	00032	MOVL	STATUS, R0	:
					RET		:
08	A3	0400	8F	3C	MOVZWL	#1024, CMDBUF_DSC	: 0816
0C	A3	FC00	C3	9E	MOVAB	INPUT_LINE, CMDBUF_DSC+4	:
04	A3	FC00	C3	9E	MOVAB	INPUT_LINE, INPUT_DSC+4	: 0817
			53	DD	PUSHL	R3	: 0818
			7E	D4	CLRL	-(SP)	:
		08	A3	9F	PUSHAB	CMDBUF_DSC	:

LATCP
V04-000

LAT Control Program
LCP_MAIN Main entry point

K 6
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 24
(4)

00000000G	00	03	FB	0004C	CALLS	#3, LIB\$GET_FOREIGN	:
	24	50	E9	00053	BLBC	R0, 2\$	:
		63	B5	00056	TSTW	INPUT_DSC	0819
		20	13	00058	BEQL	2\$	:
0134		C3	D4	0005A	CLRL	FLAGS	0822
0000'		CF	9F	0005E	PUSHAB	LCP_PROMPT	0823
		54	DD	00062	PUSHL	R4	:
		54	DD	00064	PUSHL	R4	:
0000G		CF	9F	00066	PUSHAB	LCP_COMMANDS	:
		53	DD	0006A	PUSHL	R3	:
	65	05	FB	0006C	CALLS	#5, CLISDCL_PARSE	:
	52	50	D0	0006F	MOVL	R0, STATUS	:
	2A	52	E9	00072	BLBC	STATUS, 3\$	0825
	66	00	FB	00075	CALLS	#0, CLISDISPATCH	0827
		25	11	00078	BRB	3\$	0829
0000'		CF	9F	0007A	PUSHAB	LCP_PROMPT	0834
		54	DD	0007E	PUSHL	R4	:
		54	DD	00080	PUSHL	R4	:
0000G		CF	9F	00082	PUSHAB	LCP_COMMANDS	:
		7E	D4	00086	CLRL	-(SP)	:
	65	05	FB	00088	CALLS	#5, CLISDCL_PARSE	:
	52	50	D0	0008B	MOVL	R0, STATUS	:
0001827A	8F	52	D1	0008E	CMPL	STATUS, #98938	0835
		08	13	00095	BEQL	3\$	:
	E0	52	E9	00097	BLBC	STATUS, 2\$	0837
	66	00	FB	0009A	CALLS	#0, CLISDISPATCH	0839
		DB	11	0009D	BRB	2\$	0837
	50	01	D0	0009F	MOVL	#1, R0	0842
		04	000A2	RET			0844

; Routine Size: 163 bytes, Routine Base: \$CODE\$ + 0000

```

712 0845 1 %SBTTL 'LCPK_FINDDRIVER Find the address of the communication area'
713 0846 1 ROUTINE LCPK_FINDDRIVER =
714 0847 1
715 0848 1 ++
716 0849 1 FUNCTIONAL DESCRIPTION:
717 0850 1
718 0851 1 Find the address of LTDRIVER and return the address as the value of
719 0852 1 this routine, or return failure.
720 0853 1
721 0854 1 FORMAL PARAMETERS:
722 0855 1
723 0856 1 NONE
724 0857 1
725 0858 1 IMPLICIT INPUTS:
726 0859 1
727 0860 1 ioc$gl_dptlist
728 0861 1
729 0862 1 IMPLICIT OUTPUTS:
730 0863 1
731 0864 1 NONE
732 0865 1
733 0866 1 ROUTINE VALUE:
734 0867 1 COMPLETION CODES:
735 0868 1
736 0869 1 address of the communication driver
737 0870 1
738 0871 1 SIDE EFFECTS:
739 0872 1
740 0873 1 NONE
741 0874 1
742 0875 1 --
743 0876 1
744 0877 2 BEGIN
745 0878 2
746 0879 2 |
747 0880 2 | obtain the address of the ltdriver dpt and return the address of
748 0881 2 | the counter block. If no ltdriver, then return zero.
749 0882 2 |
750 0883 2 |
751 0884 2 |
752 0885 2 | size and address of the driver name string
753 0886 2 |
754 0887 2 BIND
755 0888 2 | DRVNAM = DESCRIPTOR ('LTDRIVER') : BLOCK [,BYTE]
756 0889 2 |
757 0890 2 |
758 0891 2 LOCAL
759 0892 2 | DPT : REF BLOCK [,BYTE],
760 0893 2 | DPTEND : LONG
761 0894 2 |
762 0895 2 |
763 0896 2 | scan the dpt list looking for ltdriver by name.
764 0897 2 |
765 0898 2 DPT = .IOC$GL_DPTLIST;
766 0899 2 DPTEND = IOC$GL_DPTLIST;
767 0900 2 WHILE .DPT NEQ .DPTEND
768 0901 2 DO

```



```

: 769      0902 3      BEGIN
: 770      0903 3      IF
: 771      0904 3      CHSEQL
: 772      0905 3      (
: 773      0906 3      .DRVVRNAM [DSC$W_LENGTH], .DRVVRNAM [DSC$A_POINTER],
: 774      0907 3      CH$RCHAR (DPT [DPT$T_NAME]),
: 775      0908 3      DPT [DPT$T_NAME] +1,
: 776      0909 3      0
: 777      0910 3      )
: 778      0911 3      THEN
: 779      0912 4      BEGIN
: 780      0913 4      :
: 781      0914 4      address of the counters is found in the longword
: 782      0915 4      preceeding the vector.
: 783      0916 4      :
: 784      0917 5      IF (LCP$ _AREA = .( (.DPT + .DPT [DPT$W_VECTOR]) - 4 ) )
: 785      0918 4      LSS 0
: 786      0919 4      THEN
: 787      0920 4      RETURN SUCCESS
: 788      0921 4      ELSE
: 789      0922 4      RETURN LCP$_NOTINITED
: 790      0923 4      END
: 791      0924 3      :
: 792      0925 3      DPT = .DPT [DPT$_FLINK]
: 793      0926 3      END
: 794      0927 2      :
: 795      0928 2      RETURN LCP$_NOTLOADED
: 796      0929 2
: 797      0930 1      END;
```

```

                                .PSECT $SPLIT$,NOWRT,NOEXE,2
52 45 56 49 52 44 54 4C 00C9C P.AET: .ASCII \LTDRIVER\
                                00000008 00CA4 P.AES: .LONG 8
                                00000000 00CA8 .ADDRESS P.AET
                                DRVVRNAM= P.AES
```

```

                                .PSECT $CODE$,NOWRT,2
                                007C 00000 LCPK_FINDDRIVER:
56 00000000G 00 9E 00002 .WORD Save R2,R3,R4,R5,R6 : 0846
54 66 D0 00009 MOVAB IOC$GL_DPTLIST, R6 : 0898
55 66 9E 0000C MOVL IOC$GL_DPTLIST, DPT : 0899
55 54 D1 0000F 1$: CMPL DPT, DPTEND : 0900
31 13 00012 BEQL 4$ : 0907
50 20 A4 9A 00014 MOVZBL 32(DPT), R0 : 0917
00 0000 DF 0000 CF 2D 00018 CMPC5 DRVVRNAM, @DRVVRNAM+4, #0, R0, 33(DPT)
21 A4 00021 BNEQ 3$
50 1E A4 3C 00025 MOVZWL 30(DPT), R0
0000 CF FC A044 9F 00029 PUSHAB -4(R0)[DPT]
9E D0 0002D MOVL @ (SP)+, LCP$_AREA
```

LATCP
V04-000

LAT Control Program
LCPK_FINDDRIVER

Find the address of the commun

N 6
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 27
(5)

50	04	18	00032	BGEQ	2\$	:	0918
	01	D0	00034	MOVL	#1, R0	:	0922
		04	00037	RET		:	
50	00000000G	8F	D0 00038	2\$:	MOVL	#LCPS_NOTINITED, R0	:
		04	0003F	RET		:	
54		64	D0 00040	3\$:	MOVL	(DPT), DPT	:
		CA	11 00043	BRB	1\$	:	0925
50	00000000G	8F	D0 00045	4\$:	MOVL	#LCPS_NOTLOADED, R0	:
		04	0004C	RET		:	0930

; Routine Size: 77 bytes, Routine Base: \$CODE\$ + 00A3

LAT
V04

; R


```

799 0931 1 %SBTTL 'LCP_QUAL Process multicast qualifiers'
800 0932 1 ROUTINE LCP_QUAL =
801 0933 1
802 0934 1 ++
803 0935 1 FUNCTIONAL DESCRIPTION:
804 0936 1
805 0937 1 Process qualifiers for the multicast message. Convert the qualifiers
806 0938 1 to descriptors to set in the driver.
807 0939 1
808 0940 1 FORMAL PARAMETERS:
809 0941 1
810 0942 1 NONE
811 0943 1
812 0944 1 IMPLICIT INPUTS:
813 0945 1
814 0946 1 cli processed qualifiers
815 0947 1
816 0948 1 IMPLICIT OUTPUTS:
817 0949 1
818 0950 1 qualifier descriptors
819 0951 1
820 0952 1 ROUTINE VALUE:
821 0953 1 COMPLETION CODES:
822 0954 1
823 0955 1 errors signalled
824 0956 1
825 0957 1 SIDE EFFECTS:
826 0958 1
827 0959 1 NONE
828 0960 1
829 0961 1 --
830 0962 1
831 0963 2 BEGIN
832 0964 2
833 0965 2 LOCAL
834 0966 2 BITVALUE,
835 0967 2 PTR : REF VECTOR [,BYTE],
836 0968 2 DSC : BLOCK [DSC$K_Z_BLN, BYTE],
837 0969 2 STATUS
838 0970 2 :
839 0971 2
840 0972 2
841 0973 2 ! Convert the name to a string and require that we get something
842 0974 2
843 0975 2 IF NOT LCP_GETSTR (QUAL_NAME, NAMEDSC, GHBSK_NAMELEN, NAME,
844 0976 2 -%ASCID 'SYSSNODE', SYIS_NODENAME)
845 0977 2 OR
846 0978 2 NOT LCP_FIXNAME (NAMEDSC)
847 0979 2 THEN
848 0980 2 BEGIN
849 0981 2 SIGNAL (LCP$_IVQUAL, 2, QUAL_NAME, NAMEDSC);
850 0982 2 RETURN FAILURE
851 0983 2 END
852 0984 2 ;
853 0985 2
854 0986 2 !
855 0987 2 ! The node id is optional
```



```

856 0988 2 !
857 0989 2 LCP_GETSTR (QUAL_NODEID, NODE_IDDSC, GHBSK_IDLEN, NODE_ID,
858 0990 2 %ASCID 'SYSS$ANNOUNCE', 0);
859 0991 2
860 0992 2
861 0993 2 ! The service name is optional
862 0994 2
863 0995 2 IF LCP_GETSTR (QUAL_SERV_NAME, SERVDSC, GHBSK_NAMELEN, SERV,
864 0996 2 %ASCID 'SYSS$SERVICE_NAME', 0)
865 0997 2 THEN
866 0998 2 BEGIN
867 0999 2 IF NOT LCP_FIXNAME (SERVDSC)
868 1000 2 THEN
869 1001 2 BEGIN
870 1002 2 SIGNAL (LCP$ IVQUAL, 2, QUAL_SERV_NAME, SERVDSC);
871 1003 2 RETURN FAILURE
872 1004 2 END
873 1005 2 END
874 1006 2 ;
875 1007 2
876 1008 2 ! The service id string is optional too
877 1009 2
878 1010 2 LCP_GETSTR (QUAL_ID, IDDSC, GHBSK_IDLEN, ID, %ASCID 'SYSS$ANNOUNCE', 0);
879 1011 2
880 1012 2
881 1013 2 ! Convert the groups as a list of items
882 1014 2
883 1015 2 $SETDSC (DSC, NUMLN, NUMLST);
884 1016 2 STATUS = CLISGET_VALUE (QUAL_GROUPS, DSC);
885 1017 2
886 1018 2 ! If there is no QUALIFIER for GROUPS,
887 1019 2 ! Then we will default to enable group code 0
888 1020 2
889 1021 2 IF NOT .STATUS
890 1022 2 THEN
891 1023 2 GROUPS = 1
892 1024 2 ELSE
893 1025 2 BEGIN
894 1026 2 GROUPS = 0;
895 1027 2 WHILE .STATUS
896 1028 2 DO
897 1029 2 BEGIN
898 1030 2 STR$TRIM (DSC, DSC, DSC [DSC$W_LENGTH] );
899 1031 2 STATUS = OTSS$CVT_TI_L (DSC, BITVALUE);
900 1032 2 IF NOT .STATUS
901 1033 2 THEN
902 1034 2 BEGIN
903 1035 2 SIGNAL (LCP$ IVQUAL, 2, QUAL_GROUPS, DSC, .STATUS);
904 1036 2 RETURN FAILURE
905 1037 2 END
906 1038 2
907 1039 2 !
908 1040 2 IF
909 1041 2 .BITVALUE LSS 0
910 1042 2 OR
911 1043 2 .BITVALUE GTR 255
912 1044 2 THEN
```



```
: 913      1045 5      BEGIN
: 914      1046 5      SIGNAL (LCPS_IVQUAL, 2, QUAL_GROUPS, DSC);
: 915      1047 5      RETURN FAILURE
: 916      1048 5      END
: 917      1049 4      :
: 918      1050 4      GROUPS [0, .BITVALUE, 1, 0] = TRUE;
: 919      1051 4      $SETDSC (DSC, NUMLN, NUMLST);
: 920      1052 4      STATUS = CLISGET_VALUE (QUAL_GROUPS, DSC);
: 921      1053 4      END
: 922      1054 3      :
: 923      1055 3      END
: 924      1056 3      :
: 925      1057 2      :
: 926      1058 2      :
: 927      1059 2      : The host node rating
: 928      1060 2      :
: 929      1061 2      $SETDSC (NUMLSTDSC, NUMLN, NUMLST);
: 930      1062 2      IF CLISGET_VALUE (QUAL_RATING, NUMLSTDSC)
: 931      1063 2      THEN
: 932      1064 3      BEGIN
: 933      1065 3      STR$TRIM (NUMLSTDSC, NUMLSTDSC, NUMLSTDSC [DSC$W_LENGTH] );
: 934      1066 3      STATUS = OTSS$CVT_TI_L (NUMLSTDSC, RATING);
: 935      1067 3      IF NOT .STATUS
: 936      1068 3      OR
: 937      1069 4      (.RATING GEQU 255)
: 938      1070 3      THEN
: 939      1071 4      BEGIN
: 940      1072 4      SIGNAL (LCPS_IVQUAL, 2, QUAL_RATING, NUMLSTDSC, .STATUS);
: 941      1073 4      RETURN FAILURE
: 942      1074 4      END
: 943      1075 3      END
: 944      1076 2      :
: 945      1077 2      :
: 946      1078 2      :
: 947      1079 2      : The service name rating
: 948      1080 2      :
: 949      1081 2      $SETDSC (NUMLSTDSC, NUMLN, NUMLST);
: 950      1082 2      IF CLISGET_VALUE (QUAL_SERV_RATING, NUMLSTDSC)
: 951      1083 2      THEN
: 952      1084 3      BEGIN
: 953      1085 3      STR$TRIM (NUMLSTDSC, NUMLSTDSC, NUMLSTDSC [DSC$W_LENGTH] );
: 954      1086 3      STATUS = OTSS$CVT_TI_L (NUMLSTDSC, SERV_RATING);
: 955      1087 3      IF NOT .STATUS
: 956      1088 3      OR
: 957      1089 4      (.SERV_RATING GEQU 255)
: 958      1090 3      THEN
: 959      1091 4      BEGIN
: 960      1092 4      SIGNAL (LCPS_IVQUAL, 2, QUAL_SERV_RATING, NUMLSTDSC, .STATUS);
: 961      1093 4      RETURN FAILURE
: 962      1094 4      END
: 963      1095 3      END
: 964      1096 2      :
: 965      1097 2      :
: 966      1098 2      :
: 967      1099 2      : Multicast timer setting
: 968      1100 2      :
: 969      1101 2      $SETDSC (NUMLSTDSC, NUMLN, NUMLST);
```

```

: 970      1102 2      IF CLISGET_VALUE (QUAL_MCAST, NUMLSTDSC)
: 971      1103 2      THEN
: 972      1104 2      BEGIN
: 973      1105 2      STRSTRIM (NUMLSTDSC, NUMLSTDSC, NUMLSTDSC [DSCSW_LENGTH] );
: 974      1106 2      STATUS = OTSSCVT_TI_L (NUMLSTDSC, MCASTIMR);
: 975      1107 2      IF NOT .STATUS
: 976      1108 2      THEN
: 977      1109 2      BEGIN
: 978      1110 2      SIGNAL (LCPS_IVQUAL, 2, QUAL_MCAST, NUMLSTDSC, .STATUS);
: 979      1111 2      RETURN FAILURE
: 980      1112 2      END
: 981      1113 2      ELSE
: 982      1114 2      IF (.MCASTIMR GTRU 255)
: 983      1115 2      OR
: 984      1116 2      (.MCASTIMR LSS 10)
: 985      1117 2      THEN
: 986      1118 2      BEGIN
: 987      1119 2      SIGNAL (LCPS_IVQUAL, 2, QUAL_MCAST, NUMLSTDSC);
: 988      1120 2      RETURN FAILURE
: 989      1121 2      END
: 990      1122 2      END
: 991      1123 2      ;
: 992      1124 2      SUCCESS
: 993      1125 2      ;
: 994      1126 2      ;
: 995      1127 2      ;
: 996      1128 1      END;
```

```

                                .PSECT $SPLITS,NOWRT,NOEXE,2
                                45 44 4F 4E 24 53 59 53 00CAC P.AEV: .ASCII \SYSSNODE\
                                010E0008 00CB4 P.AEU: .LONG 17694728
                                00000000 00CB8 .ADDRESS P.AEV
                                45 43 4E 55 4F 4E 4E 41 24 53 59 53 00CBC P.AEX: .ASCII \SYSSANNOUNCE\
                                010E000C 00CC8 P.AEW: .LONG 17694732
                                00000000 00CCC .ADDRESS P.AEX
                                4D 41 4E 5F 45 43 49 56 52 45 53 24 53 59 53 00CD0 P.AEZ: .ASCII \SYSSSERVICE_NAME\
                                00CDF
                                010E0010 00CE0 P.AEY: .LONG 17694736
                                00000000 00CE4 .ADDRESS P.AEZ
                                45 43 4E 55 4F 4E 4E 41 24 53 59 53 00CE8 P.AFB: .ASCII \SYSSANNOUNCE\
                                010E000C 00CF4 P.AFA: .LONG 17694732
                                00000000 00CF8 .ADDRESS P.AFB
```

```

                                .PSECT $CODE$,NOWRT,2
                                OFFC 00000 LCP_QUAL:
                                5B 00000000G 00 9E 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11
                                5A 00000000G 8F D0 00009 MOVAB LIB$SIGNAL, R11
                                59 00000000G CF 9E 00010 MOVL #LCPS_IVQUAL, R10
                                58 00000000G 00 9E 00015 MOVAB LCP_GETSTR, R9
                                57 00000000G 00 9E 0001C MOVAB OTSSCVT TI_L, R8
                                MOVAB STRSTRIM, R7
```

0932

56	00000000G	00	9E	00023	MOVAB	CLISGET VALUE, R6	
55	0000'	CF	9E	0002A	MOVAB	QUAL_GROUPS, R5	
54	0000'	CF	9E	0002F	MOVAB	NUMLSTDSC, R4	
5E		0C	C2	00034	SUBL2	#12, SP	
7E	10D9	8F	3C	00037	MOVZWL	#4313, -(SP)	0975
	0C54	C5	9F	0003C	PUSHAB	P.AEU	
	FF30	C4	9F	00040	PUSHAB	NAME	
		10	DD	00044	PUSHL	#16	
	FF40	C4	9F	00046	PUSHAB	NAMEDSC	
	D0	A5	9F	0004A	PUSHAB	QUAL_NAME	
69		06	FB	0004D	CALLS	#6, LCP_GETSTR	
0C		50	E9	00050	BLBC	R0, 1\$	
	FF40	C4	9F	00053	PUSHAB	NAMEDSC	0978
0000V	CF	01	FB	00057	CALLS	#1, LCP_FIXNAME	
09		50	E8	0005C	BLBS	R0, 2\$	
	FF40	C4	9F	0005F	PUSHAB	NAMEDSC	0981
	D0	A5	9F	00063	PUSHAB	QUAL_NAME	
		43	11	00066	BRB	3\$	
		7E	D4	00068	CLRL	-(SP)	0989
	0C68	C5	9F	0006A	PUSHAB	P.AEW	
	A8	A4	9F	0006E	PUSHAB	NODE_ID	
7E	40	8F	9A	00071	MOVZBL	#64, -(SP)	
	E8	A4	9F	00075	PUSHAB	NODE_IDDSC	
	0120	C5	9F	00078	PUSHAB	QUAL_NODEID	
69		06	FB	0007C	CALLS	#6, LCP_GETSTR	
	0C80	7E	D4	0007F	CLRL	-(SP)	0995
	FF48	C5	9F	00081	PUSHAB	P.AEY	
		C4	9F	00085	PUSHAB	SERV	
		10	DD	00089	PUSHL	#16	
	FF58	C4	9F	0008B	PUSHAB	SERVDSC	
	3C	A5	9F	0008F	PUSHAB	QUAL_SERV_NAME	
69		06	FB	00092	CALLS	#6, LCP_GETSTR	
15		50	E9	00095	BLBC	R0, 4\$	
	FF58	C4	9F	00098	PUSHAB	SERVDSC	0999
0000V	CF	01	FB	0009C	CALLS	#1, LCP_FIXNAME	
09		50	E8	000A1	BLBS	R0, 4\$	
	FF58	C4	9F	000A4	PUSHAB	SERVDSC	1002
	3C	A5	9F	000A8	PUSHAB	QUAL_SERV_NAME	
		78	11	000AB	BRB	10\$	
		7E	D4	000AD	CLRL	-(SP)	1011
	0C94	C5	9F	000AF	PUSHAB	P.AFA	
	FF60	C4	9F	000B3	PUSHAB	ID	
7E	40	8F	9A	000B7	MOVZBL	#64, -(SP)	
	A0	A4	9F	000BB	PUSHAB	IDDSC	
	F0	A5	9F	000BE	PUSHAB	QUAL_ID	
69		06	FB	000C1	CALLS	#6, LCP_GETSTR	
04	AE	10	D0	000C4	MOVL	#16, DSC	1016
08	AE	A4	9E	000C8	MOVAB	NUMLST, DSC+4	
		AE	9F	000CD	PUSHAB	DSC	1017
		55	DD	000D0	PUSHL	R5	
66		02	FB	000D2	CALLS	#2, CLISGET_VALUE	
53		50	D0	000D5	MOVL	R0, STATUS	
06		53	E8	000D8	BLBS	STATUS, 5\$	1022
08	A4	01	D0	000DB	MOVL	#1, GROUPS	1024
		62	11	000DF	BRB	13\$	
	08	A4	D4	000E1	CLRL	GROUPS	1027
5C		53	E9	000E4	BLBC	STATUS, 13\$	1028

		04	AE	9F	000E7	PUSHAB	DSC	:	1031
		08	AE	9F	000EA	PUSHAB	DSC	:	
		0C	AE	9F	000ED	PUSHAB	DSC	:	
	67		03	FB	000F0	CALLS	#3, STR\$TRIM	:	
			5E	DD	000F3	PUSHL	SP	:	1032
		08	AE	9F	000F5	PUSHAB	DSC	:	
	68		02	FB	000F8	CALLS	#2, OT\$SCVT_TI_L	:	
	53		50	D0	000FB	MOVL	R0, STATUS	:	
	11		53	E8	000FE	BLBS	STATUS, 8\$	:	1033
			53	DD	00101	PUSHL	STATUS	:	1036
		08	AE	9F	00103	PUSHAB	DSC	:	
			55	DD	00106	PUSHL	R5	:	
			02	DD	00108	PUSHL	#2	:	
			5A	DD	0010A	PUSHL	R10	:	
	6B		05	FB	0010C	CALLS	#5, LIB\$SIGNAL	:	
			00FD	31	0010F	BRW	22\$	:	1037
	52		6E	D0	00112	MOVL	BITVALUE, R2	:	1041
			09	19	00115	BLSS	9\$	:	
000000FF	8F		52	D1	00117	CMPL	R2, #255	:	1043
			08	15	0011E	BLEQ	11\$	:	
		04	AE	9F	00120	PUSHAB	DSC	:	1046
			55	DD	00123	PUSHL	R5	:	
			00E0	31	00125	BRW	21\$	:	
00	08	A4	52	E2	00128	BBSS	R2, GROUPS, 12\$	:	1050
	04	AE	10	D0	0012D	MOVL	#16, DSC	:	1051
	08	AE	F0	A4	9E	MOVAB	NUMLST, DSC+4	:	
			04	AE	9F	PUSHAB	DSC	:	1052
				55	DD	PUSHL	R5	:	
	66		02	FB	0013B	CALLS	#2, CLISGET_VALUE	:	
	53		50	D0	0013E	MOVL	R0, STATUS	:	
			A1	11	00141	BRB	6\$	:	1028
	64		10	D0	00143	MOVL	#16, NUMLSTDSC	:	1061
04	A4	F0	A4	9E	00146	MOVAB	NUMLST, NUMLSTDSC+4	:	
			54	DD	0014B	PUSHL	R4	:	1062
		10	A5	9F	0014D	PUSHAB	QUAL_RATING	:	
	66		02	FB	00150	CALLS	#2, CLISGET_VALUE	:	
	2A		50	E9	00153	BLBC	R0, 16\$	:	
			54	DD	00156	PUSHL	R4	:	1065
			54	DD	00158	PUSHL	R4	:	
			54	DD	0015A	PUSHL	R4	:	
	67		03	FB	0015C	CALLS	#3, STR\$TRIM	:	
		28	A4	9F	0015F	PUSHAB	RATING	:	1066
			54	DD	00162	PUSHL	R4	:	
	68		02	FB	00164	CALLS	#2, OT\$SCVT_TI_L	:	
	53		50	D0	00167	MOVL	R0, STATUS	:	
	0A		53	E9	0016A	BLBC	STATUS, 14\$	:	1067
000000FF	8F	28	A4	D1	0016D	CMPL	RATING, #255	:	1069
			09	1F	00175	BLSSU	16\$	:	
			53	DD	00177	PUSHL	STATUS	:	1072
			54	DD	00179	PUSHL	R4	:	
		10	A5	9F	0017B	PUSHAB	QUAL_RATING	:	
			88	11	0017E	BRB	7\$	:	
	64		10	D0	00180	MOVL	#16, NUMLSTDSC	:	1081
04	A4	F0	A4	9E	00183	MOVAB	NUMLST, NUMLSTDSC+4	:	
			54	DD	00188	PUSHL	R4	:	1082
		28	A5	9F	0018A	PUSHAB	QUAL_SERV_RATING	:	
	66		02	FB	0018D	CALLS	#2, CLISGET_VALUE	:	

	2A	50	E9	00190	BLBC	R0, 18\$	
		54	DD	00193	PUSHL	R4	1085
		54	DD	00195	PUSHL	R4	
		54	DD	00197	PUSHL	R4	
	67	03	FB	00199	CALLS	#3, STR\$TRIM	
		2C	A4	9F 0019C	PUSHAB	SERV\$RATING	1086
		54	DD	0019F	PUSHL	R4	
	68	02	FB	001A1	CALLS	#2, OTS\$CVT_TI_L	
	53	50	D0	001A4	MOVL	R0, STATUS	
	0A	53	E9	001A7	BLBC	STATUS, 17\$	1087
000000FF	8F	2C	A4	D1 001AA	CMPL	SERV\$RATING, #255	1089
		09	1F	001B2	BLSSU	18\$	
		53	DD	001B4	PUSHL	STATUS	1092
		54	DD	001B6	PUSHL	R4	
		28	A5	9F 001B8	PUSHAB	QUAL_SERV_RATING	
		C1	11	001BB	BRB	15\$	
	64	10	D0	001BD	MOVL	#16, NUMLSTDSC	1101
04	A4	F0	A4	9E 001C0	MOVAB	NUMLST, NUMLSTDSC+4	
			54	DD 001C5	PUSHL	R4	1102
		00CC	C5	9F 001C7	PUSHAB	QUAL_MCAST	
	66	02	FB	001CB	CALLS	#2, CLIS\$GET_VALUE	
	41	50	E9	001CE	BLBC	R0, 23\$	
		54	DD	001D1	PUSHL	R4	1105
		54	DD	001D3	PUSHL	R4	
		54	DD	001D5	PUSHL	R4	
	67	03	FB	001D7	CALLS	#3, STR\$TRIM	
		38	A4	9F 001DA	PUSHAB	MCASTIMR	1106
		54	DD	001DD	PUSHL	R4	
	68	02	FB	001DF	CALLS	#2, OTS\$CVT_TI_L	
	53	50	D0	001E2	MOVL	R0, STATUS	
	0A	53	E8	001E5	BLBS	STATUS, 19\$	1107
		53	DD	001E8	PUSHL	STATUS	1110
		54	DD	001EA	PUSHL	R4	
		00CC	C5	9F 001EC	PUSHAB	QUAL_MCAST	
		8C	11	001F0	BRB	15\$	
000000FF	8F	38	A4	D1 001F2	CMPL	MCASTIMR, #255	1114
			06	1A 001FA	BGTRU	20\$	
	0A	38	A4	D1 001FC	CMPL	MCASTIMR, #10	1116
			10	18 00200	BGEQ	23\$	
			54	DD 00202	PUSHL	R4	1119
		00CC	C5	9F 00204	PUSHAB	QUAL_MCAST	
			02	DD 00208	PUSHL	#2	
			5A	DD 0020A	PUSHL	R10	
	6B	04	FB	0020C	CALLS	#4, LIB\$SIGNAL	
		50	D4	0020F	CLRL	R0	1120
			04	00211	RET		
	50	01	D0	00212	MOVL	#1, R0	1128
			04	00215	RET		

; Routine Size: 534 bytes, Routine Base: \$CODE\$ + 00F0

```

998 1129 1 %SBTTL 'LCP_GETSTR Process a string value for a qualifier'
999 1130 1 ROUTINE LCP_GETSTR (QUALDSC, DSC, SIZE, STR, LOGDSC, SYID) =
1000 1131 1
1001 1132 1 **
1002 1133 1 FUNCTIONAL DESCRIPTION:
1003 1134 1
1004 1135 1 Accept a string value for a qualifier and translate it in the
1005 1136 1 case its a logical name. If there is not return value from the
1006 1137 1 call to CLI then use the LOGDSC to get our own interpretation.
1007 1138 1 (This should be done by the CLI, but it doesn't work that way.)
1008 1139 1
1009 1140 1 Also, if the LOGDSC doesn't translate, then attempt to use the
1010 1141 1 GETSYI parameter to retrieve a value.
1011 1142 1
1012 1143 1 FORMAL PARAMETERS:
1013 1144 1
1014 1145 1     qualdsc      address of descriptor of the qualifier string
1015 1146 1     dsc          descriptor to leave string
1016 1147 1     size         size of string
1017 1148 1     str          string
1018 1149 1     logdsc       descriptor of logical name for defaulting
1019 1150 1     syid         system id parameter for $GETSYI call
1020 1151 1
1021 1152 1 IMPLICIT INPUTS:
1022 1153 1
1023 1154 1     NONE
1024 1155 1
1025 1156 1 IMPLICIT OUTPUTS:
1026 1157 1
1027 1158 1     NONE
1028 1159 1
1029 1160 1 ROUTINE VALUE:
1030 1161 1 COMPLETION CODES:
1031 1162 1
1032 1163 1     success if sring is present, failure if not, other errors signalled
1033 1164 1
1034 1165 1 SIDE EFFECTS:
1035 1166 1
1036 1167 1     NONE
1037 1168 1
1038 1169 1 --
1039 1170 1
1040 1171 2 BEGIN
1041 1172 2
1042 1173 2 LITERAL
1043 1174 2     ITEM_SIZE = 16
1044 1175 2 ;
1045 1176 2
1046 1177 2 MACRO
1047 1178 2     ITM_LEN = 0,0,16,0%,
1048 1179 2     ITM_ID  = 2,0,16,0%,
1049 1180 2     ITM_RETBUF = 4,0,32,0%,
1050 1181 2     ITM_RETLN = 8,0,32,0%,
1051 1182 2     ITM_END  = 12,0,32,0%
1052 1183 2 ;
1053 1184 2
1054 1185 2 MAP
```



```

: 1055      1186 2      STR : REF VECTOR [,BYTE],
: 1056      1187 2      DSC : REF BLOCK [,BYTE],
: 1057      1188 2      LOGDSC : REF BLOCK [,BYTE],
: 1058      1189 2      SYID
: 1059      1190 2      ;
: 1060      1191 2
: 1061      1192 2      LOCAL
: 1062      1193 2      RDSC      : BLOCK [DSC$K_Z BLN, BYTE],
: 1063      1194 2      PTR      : REF VECTOR [,BYTE],
: 1064      1195 2      ITMLST   : BLOCK [ITEM SIZE, BYTE],
: 1065      1196 2      IOSB     : VECTOR [4, WORD],
: 1066      1197 2      RET_LEN,
: 1067      1198 2      LEN,
: 1068      1199 2      STATUS
: 1069      1200 2      ;
: 1070      1201 2
: 1071      1202 2      ! Obtain the string
: 1072      1203 2      !
: 1073      1204 2      $SETDSC (DSC, .SIZE, .STR);
: 1074      1205 2      STATUS = CL$GET_VALUE (.QUALDSC, .DSC);
: 1075      1206 2      IF NOT .STATUS
: 1076      1207 2      THEN
: 1077      1208 2      BEGIN
: 1078      1209 2      $SETDSC (DSC, .SIZE, .STR);          ! Re-set descriptor
: 1079      1210 2
: 1080      1211 2      ! If there is no logical name or translation OR no system parameter,
: 1081      1212 2      ! then return failure.
: 1082      1213 2      !
: 1083      1214 2      !
: 1084      1215 4      IF NOT (.LOGDSC NEQ 0
: 1085      1216 4      AND
: 1086      1217 5      (STATUS = $TRNLOG
: 1087      1218 5      (LOGNAM = .LOGDSC, RSLLEN = DSC[DSC$W_LENGTH],
: 1088      1219 4      RSLBUF = .DSC)))
: 1089      1220 3      OR
: 1090      1221 3      .STATUS EQL SSS_NOTRAN
: 1091      1222 3      THEN
: 1092      1223 3      IF .SYID NEQ 0
: 1093      1224 3      THEN
: 1094      1225 4      BEGIN
: 1095      1226 4      ITMLST [ITM_LEN] = .SIZE;
: 1096      1227 4      ITMLST [ITM_ID] = .SYID;
: 1097      1228 4      ITMLST [ITM_RETBUF] = .STR;
: 1098      1229 4      ITMLST [ITM_RETLN] = DSC [DSC$W_LENGTH];
: 1099      1230 4      ITMLST [ITM_END] = 0;      ! End of LIST.
: 1100      1231 5      IF NOT (STATUS = $GETSYIW ( ITMLST = ITMLST,
: 1101      1232 6      IOSB = IOSB)
: 1102      1233 5      OR
: 1103      1234 5      .IOSB [0] )
: 1104      1235 4      THEN
: 1105      1236 5      BEGIN
: 1106      1237 5      DSC [DSC$W_LENGTH] = 0;
: 1107      1238 5      RETURN FAILURE
: 1108      1239 5      END
: 1109      1240 4      END
: 1110      1241 3      ELSE
: 1111      1242 4      BEGIN
```

```

1112 1243 4 DSC [DSC$W_LENGTH] = 0;
1113 1244 4 RETURN FAILURE
1114 1245 4 END
1115 1246 4
1116 1247 4 END ;
1117 1248 4 ;
1118 1249 4
1119 1250 4
1120 1251 4 Translate the name in case its a logical name
1121 1252 4
1122 1253 4 $SETDSC (RDSC, .SIZE, .STR);
1123 1254 4 DECR DCX FROM 10 TO 1
1124 1255 4 DO
1125 1256 4 BEGIN
1126 1257 4 STR$TRIM (.DSC, .DSC, DSC [DSC$W_LENGTH] );
1127 1258 4 STATUS = $TRNLOG
1128 1259 4 (LOGNAM = .DSC, RSLLEN = RDSC [DSC$W_LENGTH], RSLBUF = RDSC);
1129 1260 4 IF .STATUS EQL $$$_NOTRAN
1130 1261 4 THEN
1131 1262 4 EXITLOOP
1132 1263 4 ;
1133 1264 4 IF NOT .STATUS
1134 1265 4 THEN
1135 1266 4 BEGIN
1136 1267 4 SIGNAL (LCPS$ IVQUAL, 2, .QUALDSC, .DSC, .STATUS);
1137 1268 4 RETURN FAILURE
1138 1269 4 END
1139 1270 4 END
1140 1271 4 ;
1141 1272 4
1142 1273 4
1143 1274 4 Trim off the trailing spaces and look for an indirect or untranslated
1144 1275 4 logical name. If either, then just return none available.
1145 1276 4
1146 1277 4 STR$TRIM (.DSC, .DSC, DSC [DSC$W_LENGTH] );
1147 1278 4 PTR = .DSC [DSC$A_POINTER];
1148 1279 4 LEN = .DSC [DSC$W_LENGTH];
1149 1280 4
1150 1281 4
1151 1282 4 Scan the name and translate the non-printables to spaces
1152 1283 4
1153 1284 4 DECR DCX FROM .LEN - 1 TO 0
1154 1285 4 DO
1155 1286 4 BEGIN
1156 1287 4 SELECTONE TRUE OF
1157 1288 4 SET
1158 1289 4 [.PTR [.DCX] LSS ' ' ] : PTR [.DCX] = ' ';
1159 1290 4 [OTHERWISE] : 0;
1160 1291 4 TES
1161 1292 4 END
1162 1293 4 ;
1163 1294 4
1164 1295 4 DECR DCX FROM .LEN TO 0
1165 1296 4 DO
1166 1297 4 BEGIN
1167 1298 4 IF .PTR [0] EQL ' '
1168 1299 4 THEN

```



```
: 1169      1300  4      BEGIN
: 1170      1301  4      PTR = DSC [DSC$A_POINTER] = .PTR + 1;
: 1171      1302  4      LEN = DSC [DSC$W_LENGTH] = .LEN - 1
: 1172      1303  4      END
: 1173      1304  3      END
: 1174      1305  2      ;
: 1175      1306  2      IF .PTR [0] EQL 'a'
: 1176      1307  2      OR
: 1177      1308  2      CH$EQL (MIN (4, .LEN), .PTR, 4, UPLIT ('SYSS'))
: 1178      1309  2      THEN
: 1179      1310  2      BEGIN
: 1180      1311  3      DSC [DSC$W_LENGTH] = MIN (.QUAL_NONE [DSC$W_LENGTH], .SIZE);
: 1181      1312  3      DSC [DSC$A_POINTER] = .QUAL_NONE [DSC$A_POINTER];
: 1182      1313  3      RETURN FAILURE
: 1183      1314  3      END
: 1184      1315  2      ;
: 1185      1316  2      SUCCESS
: 1186      1317  2
: 1187      1318  2
: 1188      1319  1      END;
```

.PSECT \$PLITS,NOWRT,NOEXE,2

24 53 59 53 00CFC P.AFC: .ASCII \SYSS\ ;

.EXTRN SYS\$TRNLOG, SYS\$GETSYIW

.PSECT \$CODE\$,NOWRT,2

01FC 00000 LCP_GETSTR:

58	00000000G	00	9E	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8	: 1130
57	00000000G	00	9E	00009	MOVAB	STR\$TRIM, R8	:
5E		20	C2	00010	MOVAB	SYS\$TRNLOG, R7	:
54	08	AC	7D	00013	SUBL2	#32, SP	:
64		55	3C	00017	MOVQ	DSC, R4	: 1205
56	04	A4	9E	0001A	MOVZWL	R5, (R4)	:
66	10	AC	D0	0001E	MOVAB	4(R4), R6	:
		54	DD	00022	MOVL	STR, (R6)	:
		AC	DD	00024	PUSHL	R4	: 1206
00000000G	00	02	FB	00027	PUSHL	QUALDSC	:
52		50	D0	0002E	CALLS	#2, CL\$GET_VALUE	:
63		52	E8	00031	MOVL	R0, STATUS	:
64		55	3C	00034	BLBS	STATUS, 3\$	: 1207
66	10	AC	D0	00037	MOVZWL	R5, (R4)	: 1210
	14	AC	D5	0003B	MOVL	STR, (R6)	:
		1D	13	0003E	TSTL	LOGDSC	: 1215
		7E	7C	00040	BEQL	1\$	:
		7E	D4	00042	CLRQ	-(SP)	: 1219
		54	DD	00044	CLRL	-(SP)	:
		54	DD	00046	PUSHL	R4	:
	14	AC	DD	00048	PUSHL	R4	:
67		06	FB	0004B	PUSHL	LOGDSC	:
52		50	D0	0004E	CALLS	#6, SYS\$TRNLOG	:
09		52	E9	00051	MOVL	R0, STATUS	:
					BLBC	STATUS, 1\$	:

00000629	8F	52	D1	00054	CMPL	STATUS, #1577	1221	
		3A	12	0005B	BNEQ	3\$		
		18	AC	D5	0005D	1\$: TSTL	SYID	1223
			31	13	00060	BEQL	2\$	
08	AE	55	B0	00062	MOVW	R5, ITMLST	1226	
0A	AE	18	AC	B0	00066	MOVW	SYID, ITMLST+2	1227
0C	AE	10	AC	D0	0006B	MOVL	STR, ITMLST+4	1228
10	AE	54	D0	00070	MOVL	R4, ITMLST+8	1229	
		14	AE	D4	00074	CLRL	ITMLST+12	1230
			7E	7C	00077	CLRQ	-(SP)	1232
		08	AE	9F	00079	PUSHAB	IOSB	
		14	AE	9F	0007C	PUSHAB	ITMLST	
			7E	7C	0007F	CLRQ	-(SP)	
			7E	D4	00081	CLRL	-(SP)	
00000000G	00	07	FB	00083	CALLS	#7, SYSSGETSYIW		
	52	6E	3C	0008A	MOVZWL	IOSB, STATUS	1234	
	52	50	C8	0008D	BISL2	R0, STATUS		
	04	52	E8	00090	BLBS	STATUS, 3\$	1233	
		64	B4	00093	2\$: CLRW	(R4)	1243	
		49	11	00095	BRB	5\$	1244	
18	AE	55	3C	00097	3\$: MOVZWL	R5, RDSC	1253	
1C	AE	10	AC	D0	0009B	MOVL	STR, RDSC+4	
	53	0A	D0	000A0	MOVL	#10, DCX	1254	
		54	DD	000A3	4\$: PUSHL	R4	1257	
		54	DD	000A5	PUSHL	R4		
		54	DD	000A7	PUSHL	R4		
	68	03	FB	000A9	CALLS	#3, STR\$TRIM		
		7E	7C	000AC	CLRQ	-(SP)	1259	
		7E	D4	000AE	CLRL	-(SP)		
		24	AE	9F	000B0	PUSHAB	RDSC	
		28	AE	9F	000B3	PUSHAB	RDSC	
		54	DD	000B6	PUSHL	R4		
	67	06	FB	000B8	CALLS	#6, SYS\$TRNLOG		
	52	50	D0	000BB	MOVL	R0, STATUS		
00000629	8F	52	D1	000BE	CMPL	STATUS, #1577	1260	
		1E	13	000C5	BEQL	7\$		
	18	52	E8	000C7	BLBS	STATUS, 6\$	1264	
		52	DD	000CA	PUSHL	STATUS	1267	
		54	DD	000CC	PUSHL	R4		
		04	AC	DD	000CE	PUSHL	QUALDSC	
		02	DD	000D1	PUSHL	#2		
00000000G	00	00000000G	8F	DD	000D3	PUSHL	#LCPS, IVQUAL	
		05	FB	000D9	CALLS	#5, LIB\$SIGNAL		
		73	11	000E0	5\$: BRB	15\$	1268	
	BE	53	F5	000E2	6\$: SOBGTR	DCX, 4\$	1264	
		54	DD	000E5	7\$: PUSHL	R4	1277	
		54	DD	000E7	PUSHL	R4		
		54	DD	000E9	PUSHL	R4		
	68	03	FB	000EB	CALLS	#3, STR\$TRIM		
	50	66	D0	000EE	MOVL	(R6), PTR	1278	
	52	64	3C	000F1	MOVZWL	(R4), LEN	1279	
	51	52	D0	000F4	MOVL	LEN, DCX	1289	
		0A	11	000F7	BRB	9\$		
	20	6140	91	000F9	8\$: CMPB	(DCX)[PTR], #32		
		04	1E	000FD	BGEQU	9\$		
6140		20	90	000FF	MOVB	#32, (DCX)[PTR]		
F3		51	F4	00103	9\$: SOBGEO	DCX, 8\$	1286	

LATCP
V04-000

LAT Control Program
LCP_GETSTR Process

a string value for a qualif

N 7
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 40
(7)

LAT
V04

64
65
6F
53
53
74
6F

51	01	A2	9E	00106	MOVAB	1(R2), DCX	: 1295
19		11	0010A	BRB	11\$		: 1298
20		60	91	0010C	CMPB	(PTR), #32	: 1301
14		12	0010F	BNEQ	11\$		: 1302
53	01	A0	9E	00111	MOVAB	1(R0), R3	: 1297
66		53	D0	00115	MOVL	R3, (R6)	: 1306
50		53	D0	00118	MOVL	R3, PTR	: 1308
53	FF	A2	9E	0011B	MOVAB	-1(R2), R3	: 1311
64		53	B0	0011F	MOVW	R3, (R4)	: 1312
52		53	D0	00122	MOVL	R3, LEN	: 1313
E4		51	F4	00125	SOBGEQ	DCX, 10\$	: 1319
8F	40	60	91	00128	CMPB	(PTR), #64	: 1319
		12	13	0012C	BEQL	13\$	: 1319
04		52	D1	0012E	CMPL	R2, #4	: 1319
52		03	15	00131	BLEQ	12\$	: 1319
60		04	D0	00133	MOVL	#4, R2	: 1319
		52	2D	00136	CMPC5	R2, (PTR), #0, #4, P.AFC	: 1319
	0000'	CF		0013B			: 1319
		18	12	0013E	BNEQ	16\$	: 1319
50	0000'	CF	3C	00140	MOVZWL	QUAL_NONE, R0	: 1319
55		50	D1	00145	CMPL	R0, R5	: 1319
		03	15	00148	BLEQ	14\$	: 1319
50		55	D0	0014A	MOVL	R5, R0	: 1319
64		50	B0	0014D	MOVW	R0, (R4)	: 1319
66	0000'	CF	D0	00150	MOVL	QUAL_NONE+4, (R6)	: 1319
		50	D4	00155	CLRL	R0	: 1319
		04		00157	RET		: 1319
50		01	D0	00158	MOVL	#1, R0	: 1319
		04		0015B	RET		: 1319

; Routine Size: 348 bytes, Routine Base: \$CODE\$ + 0306

```
: 1190      1320 1 %SBTTL 'LCP_FIXNAME Process a name string'
: 1191      1321 1 ROUTINE LCP_FIXNAME (DSC) =
: 1192      1322 1
: 1193      1323 1 ++
: 1194      1324 1 FUNCTIONAL DESCRIPTION:
: 1195      1325 1
: 1196      1326 1 Process a name string. Convert it to the correct form.
: 1197      1327 1
: 1198      1328 1 FORMAL PARAMETERS:
: 1199      1329 1
: 1200      1330 1 DSC Address of descriptor of string
: 1201      1331 1
: 1202      1332 1 IMPLICIT INPUTS:
: 1203      1333 1
: 1204      1334 1 NONE
: 1205      1335 1
: 1206      1336 1 IMPLICIT OUTPUTS:
: 1207      1337 1
: 1208      1338 1 NONE
: 1209      1339 1
: 1210      1340 1 ROUTINE VALUE:
: 1211      1341 1 COMPLETION CODES:
: 1212      1342 1
: 1213      1343 1 success or failure
: 1214      1344 1
: 1215      1345 1 SIDE EFFECTS:
: 1216      1346 1
: 1217      1347 1 NONE
: 1218      1348 1
: 1219      1349 1 --
: 1220      1350 1
: 1221      1351 2 BEGIN
: 1222      1352 2
: 1223      1353 2 MAP
: 1224      1354 2 DSC : REF BLOCK [,BYTE]
: 1225      1355 2 ;
: 1226      1356 2
: 1227      1357 2 LOCAL
: 1228      1358 2 LEN,
: 1229      1359 2 PTR : REF VECTOR [,BYTE],
: 1230      1360 2 STATUS
: 1231      1361 2 ;
: 1232      1362 2
: 1233      1363 2 !
: 1234      1364 2 ! Strip beginning underscores and trailing colons
: 1235      1365 2 !
: 1236      1366 2 PTR = .DSC [DSC$A_POINTER];
: 1237      1367 2 LEN = .DSC [DSC$W_LENGTH];
: 1238      1368 2 DO
: 1239      1369 3 BEGIN
: 1240      1370 3 STATUS = FALSE;
: 1241      1371 3 IF .PTR [0] EQL '_'
: 1242      1372 3 THEN
: 1243      1373 4 BEGIN
: 1244      1374 4 LEN = DSC [DSC$W_LENGTH] = .LEN - 1;
: 1245      1375 4 PTR = DSC [DSC$A_POINTER] = .PTR + 1;
: 1246      1376 4 STATUS = TRUE
```



```
: 1247      1377 4      END
: 1248      1378 3
: 1249      1379 3      IF .PTR [.LEN - 1] EQL ':'
: 1250      1380 3      THEN
: 1251      1381 4      BEGIN
: 1252      1382 4      LEN = DSC [DSC$W_LENGTH] = .LEN - 1;
: 1253      1383 4      STATUS = TRUE
: 1254      1384 4      END
: 1255      1385 3      END
: 1256      1386 2      WHILE .STATUS
: 1257      1387 2
: 1258      1388 2      !
: 1259      1389 2      ! Uppcase the string and check for legal characters
: 1260      1390 2
: 1261      1391 2      STR$UPCASE (.DSC, .DSC);
: 1262      1392 2      DECR DCX FROM .LEN - 1 TO 0
: 1263      1393 2      DO
: 1264      1394 3      BEGIN
: 1265      1395 3      SELECTONE TRUE OF
: 1266      1396 3      SET
: 1267      1397 3      [.PTR [.DCX] GEQ 'A' AND .PTR [.DCX] LEQ 'Z'] : 0;
: 1268      1398 3      [.PTR [.DCX] GEQ '0' AND .PTR [.DCX] LEQ '9'] : 0;
: 1269      1399 3      [.PTR [.DCX] EQL ' '] : 0;
: 1270      1400 3      [.PTR [.DCX] EQL '$'] : 0;
: 1271      1401 3      [OTHERWISE] : RETURN FAILURE;
: 1272      1402 3      TES
: 1273      1403 3      END
: 1274      1404 2
: 1275      1405 2
: 1276      1406 2      SUCCESS
: 1277      1407 2
: 1278      1408 1      END;
```

```
003C 00000 LCP_FIXNAME:
      50      04      AC      D0      00002      .WORD      Save R2,R3,R4,R5
      53      04      A0      D0      00006      MOVL      DSC, R0
      52      60      3C      0000A      MOVL      4(R0), PTR
      54      D4      0000D      MOVZWL      (R0), LEN
      5F      8F      63      91      0000F      CLRL      STATUS
      18      12      00013      CMPB      (PTR), #95
      51      FF      A2      9E      00015      BNEQ      2$
      60      51      B0      00019      MOVAB      -1(R2), R1
      52      51      D0      0001C      MOVW      R1, (R0)
      51      01      A3      9E      0001F      MOVL      R1, LEN
      04      A0      51      D0      00023      MOVAB      1(R3), R1
      53      51      D0      00027      MOVL      R1, 4(R0)
      54      01      D0      0002A      MOVL      R1, PTR
      3A      FF      A2      91      0002D      MOVL      #1, STATUS
      0D      12      00032      CMPB      -1(LEN)[PTR], #58
      51      FF      A2      9E      00034      BNEQ      3$
      60      51      B0      00038      MOVAB      -1(R2), R1
      52      51      D0      0003B      MOVW      R1, (R0)
      51      51      D0      0003B      MOVL      R1, LEN
```

1321
1366
1367
1370
1371
1374
1375
1376
1379
1382

LATCP
V04-000

LAT Control Program
LCP_FIXNAME Process a name string

D 8
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 43
(8)

	54	01	D0	0003E	MOVL	#1, STATUS	:	1383	
	C9	54	E8	00041	3\$:	BLBS	STATUS, 1\$	:	1386
		50	DD	00044		PUSHL	R0	:	1391
		50	DD	00046		PUSHL	R0	:	
	00000000G	00	02	FB	00048	CALLS	#2, STR\$UPCASE	:	
		4E	11	0004F		BRB	9\$	:	1392
	51	6243	9A	00051	4\$:	MOVZBL	(DCX)[PTR], R1	:	1397
		54	D4	00055		CLRL	R4	:	
	41	8F	51	91	00057	CMPB	R1, #65	:	
		02	1F	0005B		BLSSU	5\$	:	
		54	D6	0005D		INCL	R4	:	
		50	D4	0005F	5\$:	CLRL	R0	:	
	5A	8F	51	91	00061	CMPB	R1, #90	:	
		02	1A	00065		BGTRU	6\$	:	
		50	D6	00067		INCL	R0	:	
	55		54	D2	00069	6\$:	MCOML	R4, R5	
	50		55	CA	0006C		BICL2	R5, R0	
	01		50	D1	0006F		CMPL	R0, #1	
		2B	13	00072		BEQL	9\$	:	
		54	D4	00074		CLRL	R4	:	1398
	30		51	91	00076	CMPB	R1, #48	:	
		02	1F	00079		BLSSU	7\$	:	
		54	D6	0007B		INCL	R4	:	
		50	D4	0007D	7\$:	CLRL	R0	:	
	39		51	91	0007F	CMPB	R1, #57	:	
		02	1A	00082		BGTRU	8\$	:	
		50	D6	00084		INCL	R0	:	
	55		54	D2	00086	8\$:	MCOML	R4, R5	
	50		55	CA	00089		BICL2	R5, R0	
	01		50	D1	0008C		CMPL	R0, #1	
		0E	13	0008F		BEQL	9\$	:	
	5F	8F	51	91	00091	CMPB	R1, #95	:	1399
		08	13	00095		BEQL	9\$	:	
	24		51	91	00097	CMPB	R1, #36	:	1400
		03	13	0009A		BEQL	9\$	:	
		50	D4	0009C		CLRL	R0	:	1401
		04	0009E			RET		:	
	AF	52	F4	0009F	9\$:	SOBGEQ	DCX, 4\$	:	1394
	50	01	D0	000A2		MOVL	#1, R0	:	1408
		04	000A5			RET		:	

; Routine Size: 166 bytes, Routine Base: \$CODE\$ + 0462


```
: 1280      1409 1 %SBTTL 'LCP START Start the terminal port driver'
: 1281      1410 1 GLOBAL ROUTINE LCP_START : NOVALUE =
: 1282      1411 1
: 1283      1412 1 ++
: 1284      1413 1 FUNCTIONAL DESCRIPTION:
: 1285      1414 1
: 1286      1415 1     Handle the general START request and the START HISTORY request.
: 1287      1416 1     If it is the general start request, then set the multicast message
: 1288      1417 1     data and then start the port driver. If it's the START HISTORY
: 1289      1418 1     request, allocate the history buffer.
: 1290      1419 1
: 1291      1420 1 FORMAL PARAMETERS:
: 1292      1421 1
: 1293      1422 1     NONE
: 1294      1423 1
: 1295      1424 1 IMPLICIT INPUTS:
: 1296      1425 1
: 1297      1426 1     results from qualifier parse and address of driver comm area
: 1298      1427 1
: 1299      1428 1 IMPLICIT OUTPUTS:
: 1300      1429 1
: 1301      1430 1     NONE
: 1302      1431 1
: 1303      1432 1 ROUTINE VALUE:
: 1304      1433 1 COMPLETION CODES:
: 1305      1434 1
: 1306      1435 1     NONE
: 1307      1436 1
: 1308      1437 1 SIDE EFFECTS:
: 1309      1438 1
: 1310      1439 1     driver started
: 1311      1440 1
: 1312      1441 1 --
: 1313      1442 1
: 1314      1443 2 BEGIN
: 1315      1444 2
: 1316      1445 2 LOCAL
: 1317      1446 2     STATUS
: 1318      1447 2 ;
: 1319      1448 2
: 1320      1449 2 !
: 1321      1450 2 ! Check if this is a START HISTORY command
: 1322      1451 2 !
: 1323      1452 2 $SETDSC (SUBCMD DSC, CMDLEN, SUBCMD);
: 1324      1453 2 STATUS = CLISGET VALUE (QUAL SUBCOMMAND, SUBCMD DSC);
: 1325      1454 2 IF .STATUS EQL CCIS_ABSENT
: 1326      1455 2 THEN
: 1327      1456 3     BEGIN
: 1328      1457 3 !
: 1329      1458 3 ! Get the qualifiers and set them
: 1330      1459 3 !
: 1331      1460 4     IF NOT- (STATUS = LCP_SET ( ) )
: 1332      1461 3     THEN
: 1333      1462 3         RETURN
: 1334      1463 3 ;
: 1335      1464 3 !
: 1336      1465 3 !
```

LATCP
V04-000

LAT Control Program
LCP_START Start the terminal port driver

F 8
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 45
(9)

```

: 1337      1466 3      : Now start the port driver
: 1338      1467 3      :
: 1339      1468 4      IF (STATUS = $CMKRNL (ROUTIN = LCPK_START) )
: 1340      1469 3      THEN
: 1341      1470 4      LOG_SIGNAL (LCP$_STARTED)
: 1342      1471 3      ELSE
: 1343      1472 3      SIGNAL (LCP$_NOTSTARTED, 0, .STATUS)
: 1344      1473 3      :
: 1345      1474 3      : END
: 1346      1475 3      ELSE
: 1347      1476 3      BEGIN
: 1348      1477 3      STR$TRIM (SUBCMDDSC, SUBCMDDSC, SUBCMDDSC [DSC$_LENGTH] );
: 1349      1478 3      :
: 1350      1479 3      : Dispatch on the subcommand string
: 1351      1480 3      :
: 1352      1481 3      SELECTONE TRUE OF
: 1353      1482 3      SET
: 1354      1483 3      [ $CMPDSC (SUBCMDDSC, QUAL_HISTORY) ] :
: 1355      1484 4      (
: 1356      1485 5      IF NOT (STATUS = $CMKRNL (ROUTIN = LCPK_STARTHISTORY) )
: 1357      1486 4      THEN
: 1358      1487 4      SIGNAL (LCP$_NOSTARTHIST, 0, .STATUS)
: 1359      1488 4      ;
: 1360      1489 3      );
: 1361      1490 3      [OTHERWISE] : SIGNAL (LCP$_IVCMD);
: 1362      1491 3      :
: 1363      1492 3      TES;
: 1364      1493 3      :
: 1365      1494 3      END
: 1366      1495 3      :
: 1367      1496 2      :
: 1368      1497 1      : END;
```

			03FC 00000	.ENTRY	LCP_START, Save R2,R3,R4,R5,R6,R7,R8,R9	1410
	59	00000000G	00 9E 00002	MOVAB	LIB\$SIGNAL, R9	
	58	00000000G	00 9E 00009	MOVAB	SY\$CMKRNL, R8	
	57	0000'	CF 9E 00010	MOVAB	QUAL_HISTORY, R7	
	56	0000'	CF 9E 00015	MOVAB	SUBCMDDSC, R6	
	66		10 D0 0001A	MOVL	#16, SUBCMDDSC	1452
04	A6	F0	A6 9E 0001D	MOVAB	SUBCMD, SUBCMDDSC+4	
			56 DD 00022	PUSHL	R6	1453
		FF20	C7 9F 00024	PUSHAB	QUAL_SUBCOMMAND	
00000000G	00		02 FB 00028	CALLS	#2, CLISGET_VALUE	
	55		50 D0 0002F	MOVL	R0, STATUS	
00000000G	8F		55 D1 00032	CMPL	STATUS, #CLIS_ABSENT	1454
			3C 12 00039	BNEQ	3\$	
0000V	CF		00 FB 0003B	CALLS	#0, LCP_SET	1460
	55		50 D0 00040	MOVL	R0, STATUS	
	01		55 E8 00043	BLBS	STATUS, 1\$	
			04 00046	RET		
			7E D4 00047 1\$:	CLRL	-(SP)	1468
		0000V	CF 9F 00049	PUSHAB	LCPK_START	
	68		02 FB 0004D	CALLS	#2, SY\$CMKRNL	

	55		50	D0	00050	MOVL	R0, STATUS		
	15		55	E9	00053	BLBC	STATUS, 2\$		
		68	A7	9F	00056	PUSHAB	QUAL_LOG		1470
00000000G	00		01	FB	00059	CALLS	#1, CLISPRESNT		
	6D		50	E9	00060	BLBC	R0, 10\$		
		00000000G	8F	DD	00063	PUSHL	#LCPS_STARTED		
			62	11	00069	BRB	9\$		
			55	DD	0006B	2\$: PUSHL	STATUS		1472
			7E	D4	0006D	CLRL	-(SP)		
		00000000G	8F	DD	0006F	PUSHL	#LCPS_NOTSTARTED		
			4C	11	00075	BRB	7\$		
			56	DD	00077	3\$: PUSHL	R6		1477
			56	DD	00079	PUSHL	R6		
			56	DD	0007B	PUSHL	R6		
00000000G	00		03	FB	0007D	CALLS	#3, STRSTRIM		
	50		66	3C	00084	MOVZWL	SUBCMDSC, R0		1483
	50		67	B1	00087	CMPW	QUAL_HISTORY, R0		
			03	1E	0008A	BGEQU	4\$		
	50		67	3C	0008C	MOVZWL	QUAL_HISTORY, R0		
	03		50	B1	0008F	4\$: CMPW	R0, #3		
			03	1E	00092	BGEQU	5\$		
	50		03	D0	00094	MOVL	#3, R0		
			54	D4	00097	5\$: CLRL	R4		
50	20	04	B6	66	2D	CMPC5	SUBCMDSC, @SUBCMDSC+4, #32, R0, -		
			04	B7	0009F		@QUAL_HISTORY+4		
			02	12	000A1	BNEQ	6\$		
			54	D6	000A3	INCL	R4		
	01		54	D1	000A5	6\$: CMPL	R4, #1		
			1D	12	000A8	BNEQ	8\$		
			7E	D4	000AA	CLRL	-(SP)		1485
		0000V	CF	9F	000AC	PUSHAB	LCPK_STARTHISTORY		
	68		02	FB	000B0	CALLS	#2, SYSSCMKRN		
	55		50	D0	000B3	MOVL	R0, STATUS		
	17		55	E8	000B6	BLBS	STATUS, 10\$		
			55	DD	000B9	PUSHL	STATUS		1487
			7E	D4	000BB	CLRL	-(SP)		
		00000000G	8F	DD	000BD	PUSHL	#LCPS_NOSTARTHIST		
	69		03	FB	000C3	7\$: CALLS	#3, LIBSSIGNAL		
			04	000C6		RET			1481
		00000000G	8F	DD	000C7	8\$: PUSHL	#LCPS_IVCMD		1491
	69		01	FB	000CD	9\$: CALLS	#1, LIBSSIGNAL		
			04	000D0	10\$: RET				1497

; Routine Size: 209 bytes, Routine Base: \$CODE\$ + 0508

```
: 1370 1498 1 %SBTTL 'LCP_STOP Stop the port driver'
: 1371 1499 1 GLOBAL ROUTINE LCP_STOP : NOVALUE =
: 1372 1500 1
: 1373 1501 1 !++
: 1374 1502 1 FUNCTIONAL DESCRIPTION:
: 1375 1503 1
: 1376 1504 1 The kernel routine is called to stop the driver.
: 1377 1505 1
: 1378 1506 1 FORMAL PARAMETERS:
: 1379 1507 1
: 1380 1508 1 NONE
: 1381 1509 1
: 1382 1510 1 IMPLICIT INPUTS:
: 1383 1511 1
: 1384 1512 1 NONE
: 1385 1513 1
: 1386 1514 1 IMPLICIT OUTPUTS:
: 1387 1515 1
: 1388 1516 1 NONE
: 1389 1517 1
: 1390 1518 1 ROUTINE VALUE:
: 1391 1519 1 COMPLETION CODES:
: 1392 1520 1
: 1393 1521 1 NONE
: 1394 1522 1
: 1395 1523 1 SIDE EFFECTS:
: 1396 1524 1
: 1397 1525 1 the driver is stopped
: 1398 1526 1
: 1399 1527 1 --
: 1400 1528 1
: 1401 1529 2 BEGIN
: 1402 1530 2
: 1403 1531 2 LITERAL
: 1404 1532 2 STRSIZE = 32
: 1405 1533 2 ;
: 1406 1534 2
: 1407 1535 2 BIND
: 1408 1536 2 TTNAME = DESCRIPTOR ('TT')
: 1409 1537 2 ;
: 1410 1538 2
: 1411 1539 2 LOCAL
: 1412 1540 2 DSC : BLOCK [DSC$K_Z BLN, BYTE],
: 1413 1541 2 STR : VECTOR [STRSIZE, BYTE],
: 1414 1542 2 STATUS
: 1415 1543 2 ;
: 1416 1544 2
: 1417 1545 2 !
: 1418 1546 2 ! Check if this is a STOP HISTORY command
: 1419 1547 2 !
: 1420 1548 2 $SETDSC (SUBCMD$DSC, CMDLEN, SUBCMD);
: 1421 1549 2 STATUS = CLIS$GET VALUE (QUAL_SUBCOMMAND, SUBCMD$DSC);
: 1422 1550 2 IF .STATUS EQL C[IS_ABSENT
: 1423 1551 2 THEN
: 1424 1552 2 BEGIN
: 1425 1553 2 !
: 1426 1554 2 ! This is a general STOP command
```



```

: 1427      1555      !
: 1428      1556      !
: 1429      1557      ! We cannot stop LAT from a LAT terminal
: 1430      1558      !
: 1431      1559      $SETDSC (DSC, STRSIZE, STR);
: 1432      1560      $TRNLOG
: 1433      1561      (
: 1434      1562      LOGNAM = TTNAME,
: 1435      1563      RSLLEN = DSC [DSC$W_LENGTH],
: 1436      1564      RSLBUF = DSC
: 1437      1565      );
: 1438      1566      IF CH$EQL (4, UPLIT (%ASCII '_LTA'), 4, STR)
: 1439      1567      OR
: 1440      1568      ($CMKRNL (ROUTIN = LCPK_IS_LAT_TERMINAL) )
: 1441      1569      THEN
: 1442      1570      BEGIN
: 1443      1571      SIGNAL (LCP$_NOTFROMLAT);
: 1444      1572      RETURN
: 1445      1573      END
: 1446      1574      ;
: 1447      1575      $CMKRNL (ROUTIN = LCPK_STOPHISTORY);      ! Delete the history buffer
: 1448      1576
: 1449      1577      !
: 1450      1578      ! Stop the LTDRIVER
: 1451      1579      !
: 1452      1580      !
: 1453      1581      IF (STATUS = $CMKRNL (ROUTIN = LCPK_STOP) )
: 1454      1582      THEN
: 1455      1583      LOG_SIGNAL (LCP$_STOPPED)
: 1456      1584      ELSE
: 1457      1585      SIGNAL (LCP$_NOTSTOPPED, 0, .STATUS)
: 1458      1586      ;
: 1459      1587      END
: 1460      1588      ELSE
: 1461      1589      !
: 1462      1590      ! This is a STOP HISTORY command.
: 1463      1591      !
: 1464      1592      BEGIN
: 1465      1593      STR$TRIM (SUBCMDDSC, SUBCMDDSC, SUBCMDDSC [DSC$W_LENGTH] );
: 1466      1594      !
: 1467      1595      ! Dispatch on the subcommand string
: 1468      1596      !
: 1469      1597      SELECTONE TRUE OF
: 1470      1598      SET
: 1471      1599      [ $CMPDSC (SUBCMDDSC, QUAL_HISTORY) ] :
: 1472      1600      (
: 1473      1601      $CMKRNL (ROUTIN = LCPK_STOPHISTORY)
: 1474      1602      );
: 1475      1603
: 1476      1604      [OTHERWISE] : SIGNAL (LCP$_IVCMD);
: 1477      1605
: 1478      1606      TES;
: 1479      1607
: 1480      1608      END
: 1481      1609      ;
: 1482      1610
: 1483      1611      END;
```

	58	00000000G	00	01FC	00000	.ENTRY	LCP STOP, Save R2,R3,R4,R5,R6,R7,R8	:	1499
	57	0000'	CF	9E	00002	MOVAB	LIB\$SIGNAL, R8	:	
	56	00000000G	00	9E	00009	MOVAB	QUAL HISTORY, R7	:	
	55	0000'	CF	9E	0000E	MOVAB	SYS\$CMKRNL, R6	:	
	5E		28	C2	0001A	MOVAB	SUBCMD DSC, R5	:	
	65		10	D0	0001D	SUBL2	#40, SP	:	
04	A5	F0	A5	9E	00020	MOVL	#16, SUBCMD DSC	:	1548
			55	DD	00025	MOVAB	SUBCMD, SUBCMD DSC+4	:	
		FF20	C7	9F	00027	PUSHL	R5	:	1549
00000000G	00		02	FB	0002B	PUSHAB	QUAL SUBCOMMAND	:	
	52		50	D0	00032	CALLS	#2, CLISGET_VALUE	:	
00000000G	8F		52	D1	00035	MOVL	R0, STATUS	:	
			73	12	0003C	CMPL	STATUS, #CLIS_ABSENT	:	1550
20	AE		20	D0	0003E	BNEQ	4\$	:	
24	AE		6E	9E	00042	MOVL	#32, DSC	:	1559
			7E	7C	00046	MOVAB	STR, DSC+4	:	
			7E	D4	00048	CLRQ	-(SP)	:	1565
		2C	AE	9F	0004A	CLRL	-(SP)	:	
		30	AE	9F	0004D	PUSHAB	DSC	:	
		0C08	C7	9F	00050	PUSHAB	DSC	:	
00000000G	00		06	FB	00054	PUSHAB	TNAME	:	
	6E	0C10	C7	D1	0005B	CALLS	#6, SYS\$TRNLOG	:	
			0C	13	00060	CMPL	P.AFF, STR	:	1566
			7E	D4	00062	BEQL	1\$	:	
		0000V	CF	9F	00064	CLRL	-(SP)	:	1568
	66		02	FB	00068	PUSHAB	LC PK IS LAT TERMINAL	:	
	08		50	E9	0006B	CALLS	#2, SYS\$CMKRNL	:	
		00000000G	8F	DD	0006E	BLBC	R0, 2\$	:	
			7E	11	00074	PUSHL	#LCP\$_NOTFROMLAT	:	1571
			7E	D4	00076	BRB	9\$	:	
		0000V	CF	9F	00078	CLRL	-(SP)	:	1576
	66		02	FB	0007C	PUSHAB	LC PK STOPHISTORY	:	
		0000V	7E	D4	0007F	CALLS	#2, SYS\$CMKRNL	:	
			CF	9F	00081	CLRL	-(SP)	:	1581
	66		02	FB	00085	PUSHAB	LC PK STOP	:	
	52		50	D0	00088	CALLS	#2, SYS\$CMKRNL	:	
	15		52	E9	0008B	MOVL	R0, STATUS	:	
		68	A7	9F	0008E	BLBC	STATUS, 3\$	:	
00000000G	00		01	FB	00091	PUSHAB	QUAL LOG	:	1583
	5C		50	E9	00098	CALLS	#1, CLISPRESENT	:	
						BLBC	R0, 10\$	:	

LATCP
V04-000

LAT Control Program
LCP_STOP Stop the port driver

K 8
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 50
(10)

			00000000G	8F	DD	0009B		PUSHL	#LCP\$_STOPPED		
				51	11	000A1		BRB	9\$		
				52	DD	000A3	3\$:	PUSHL	STATUS		1585
				7E	D4	000A5		CLRL	-(SP)		
			00000000G	8F	DD	000A7		PUSHL	#LCP\$ NOTSTOPPED		
		68		03	FB	000AD		CALLS	#3, LIB\$SIGNAL		
					04	000B0		RET			1550
				55	DD	000B1	4\$:	PUSHL	R5		1593
				55	DD	000B3		PUSHL	R5		
				55	DD	000B5		PUSHL	R5		
		00000000G	00	03	FB	000B7		CALLS	#3, STR\$TRIM		
			50	65	3C	000BE		MOVZWL	SUBCMDDSC, R0		1599
			50	67	B1	000C1		CMPW	QUAL_HISTORY, R0		
				03	1E	000C4		BGEQU	5\$		
			50	67	3C	000C6		MOVZWL	QUAL_HISTORY, R0		
			03	50	B1	000C9	5\$:	CMPW	R0, #3		
				03	1E	000CC		BGEQU	6\$		
			50	03	D0	000CE		MOVL	#3, R0		
				54	D4	000D1	6\$:	CLRL	R4		
50				65	2D	000D3		CMPC5	SUBCMDDSC, @SUBCMDDSC+4, #32, R0, -		
				04	B7	000D9			@QUAL_HISTORY+4		
				02	12	000DB		BNEQ	7\$		
				54	D6	000DD		INCL	R4		
			01	54	D1	000DF	7\$:	CMPL	R4, #1		
				0A	12	000E2		BNEQ	8\$		
				7E	D4	000E4		CLRL	-(SP)		1601
			0000V	CF	9F	000E6		PUSHAB	LCPK_STOPHISTORY		
				02	FB	000EA		CALLS	#2, SYS\$CMKRNL		
					04	000ED		RET			1600
			00000000G	8F	DD	000EE	8\$:	PUSHL	#LCP\$_IVCMD		1604
				01	FB	000F4	9\$:	CALLS	#1, LIB\$SIGNAL		
				04	000F7	10\$:		RET			1611

; Routine Size: 248 bytes, Routine Base: \$CODE\$ + 05D9

```
: 1485 1612 1 %SBTTL 'LCP SET Set multicast information'
: 1486 1613 1 GLOBAL ROUTINE LCP_SET =
: 1487 1614 1
: 1488 1615 1 ++
: 1489 1616 1 FUNCTIONAL DESCRIPTION:
: 1490 1617 1
: 1491 1618 1 The qualifier information is reported and then the kernel mode
: 1492 1619 1 routine is called to set the multicast information for transmission.
: 1493 1620 1
: 1494 1621 1 FORMAL PARAMETERS:
: 1495 1622 1
: 1496 1623 1 NONE
: 1497 1624 1
: 1498 1625 1 IMPLICIT INPUTS:
: 1499 1626 1
: 1500 1627 1 results of qualifier parse
: 1501 1628 1
: 1502 1629 1 IMPLICIT OUTPUTS:
: 1503 1630 1
: 1504 1631 1 NONE
: 1505 1632 1
: 1506 1633 1 ROUTINE VALUE:
: 1507 1634 1 COMPLETION CODES:
: 1508 1635 1
: 1509 1636 1 NONE
: 1510 1637 1
: 1511 1638 1 SIDE EFFECTS:
: 1512 1639 1
: 1513 1640 1 multicast information set
: 1514 1641 1
: 1515 1642 1 --
: 1516 1643 1
: 1517 1644 1 BEGIN
: 1518 1645 2
: 1519 1646 2 LITERAL
: 1520 1647 2 STRSIZE = 512, ! Size of input and output strings
: 1521 1648 2 PRMSIZE = 24 ! Size of parameter list in bytes
: 1522 1649 2 ;
: 1523 1650 2
: 1524 1651 2 LOCAL
: 1525 1652 2 STATUS,
: 1526 1653 2 DSC : BLOCK [DSC$K_Z_BLN, BYTE],
: 1527 1654 2 STR : VECTOR [STRSIZE, BYTE],
: 1528 1655 2 P : VECTOR [PRMSIZE]
: 1529 1656 2 ;
: 1530 1657 2
: 1531 1658 2
: 1532 1659 2 Get the command qualifiers
: 1533 1660 2
: 1534 1661 3 IF NOT (STATUS = LCP_QUAL ( ) )
: 1535 1662 2 THEN
: 1536 1663 2 RETURN .STATUS
: 1537 1664 2 ;
: 1538 1665 2
: 1539 1666 2
: 1540 1667 2 Convert the group codes to an ascii string
: 1541 1668 2
```



```
: 1542      1669 2      LCP_CVTGROUPS ();
: 1543      1670 2
: 1544      1671 2
: 1545      1672 2      | Set up the parameter list and report the qualifier information
: 1546      1673 2      | that we are about to set.
: 1547      1674 2
: 1548      1675 2      PC[0] = NAMEDSC;
: 1549      1676 2      PC[1] = SERVDSC;
: 1550      1677 2      PC[2] = NODE_IDDSC;
: 1551      1678 2      PC[3] = IDDSC;
: 1552      1679 2      PC[4] = GRPDSC;
: 1553      1680 2      PC[5] = .MCASTMR;
: 1554      1681 2
: 1555      1682 2      $SETDSC (DSC, STRSIZE, STR);
: 1556      1683 2
: 1557      P 1684 2      $FAOL
: 1558      P 1685 2      (
: 1559      L 1686 2          CTRSTR = %ASCID
: 1560      L 1687 2          %STRING
: 1561      L 1688 2          (
: 1562      L 1689 2              '!/LCP Qualifier Summary!//!',
: 1563      L 1690 2              'Node name = !18<\\!AS!> Service name = !18<\\!AS!>!',
: 1564      L 1691 2              'Node Identification = \\!AS!/',
: 1565      L 1692 2              'Service Identification = \\!AS!/',
: 1566      L 1693 2              'Groups = !AS!/',
: 1567      L 1694 2              'Multicast timer = !UB seconds!/'
: 1568      P 1695 2          )
: 1569      P 1696 2          OUTBUF = DSC,
: 1570      P 1697 2          OUTLEN = DSC,
: 1571      P 1698 2          PRMLST = PC[0]
: 1572      1699 2          );
: 1573      1700 2
: 1574      1701 2      LOG_OUTPUT (DSC);
: 1575      1702 2
: 1576      1703 2      |
: 1577      1704 2      | Call the kernel mode routine to set the information in the
: 1578      1705 2      | driver.
: 1579      1706 2
: 1580      1707 3      IF NOT (STATUS = $CMKRNL (ROUTIN = LCPK_SET) )
: 1581      1708 2      THEN
: 1582      1709 2          SIGNAL (LCP$_NOTSET, 0, .STATUS)
: 1583      1710 2      ;
: 1584      1711 2
: 1585      1712 2      |
: 1586      1713 2      | LOG success if requested
: 1587      1714 2
: 1588      1715 2      LOG_SIGNAL (LCP$_SET);
: 1589      1716 2
: 1590      1717 2      SUCCESS
: 1591      1718 2
: 1592      1719 1      END;
```

.PSECT \$PLITS,NOWRT,NOEXE,2

72 65 69 66 69 6C 61 75 51 20 50 43 4C 2F 21 00D10 P.AFH: .ASCII \!/LCP Qualifier Summary!//Node name = !\ ;

LATCP
V04-000

LAT Control Program
LCP_SET Set multicast information

N 8
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 53
(11)

64	6F	4E	2F	21	2F	21	79	72	61	6D	6D	75	53	20	00D1F
65	53	20	20	20	3E	21	5C	53	41	21	5C	3C	38	31	00D2E
6F	4E	2F	21	3E	21	5C	53	41	21	5C	3C	38	31	21	00D38
53	2F	21	5C	53	41	21	5C	20	3D	20	6E	6F	69	74	00D47
53	41	21	5C	20	3D	20	6E	6F	69	74	61	63	69	66	00D54
74	20	74	73	61	63	69	74	6C	75	4D	2F	21	53	41	00D63
6F	63	65	73	20	42	55	21	20	3D	20	72	65	6D	69	00D70
															00D7F
															00D8C
															00D9B
															00DA8
															00DB7
															00DC6
															00DCC
															00DD0

010E00BB
00000000

P.AFG:

.ASCII \18<\<92>\!AS\<92>\!> Service name = \

.ASCII \!18<\<92>\!AS\<92>\!>\!/Node Identifica\

.ASCII \tion = \<92>\!AS\<92>\!>\!/Service Identi\

.ASCII \fication = \<92>\!AS\<92>\!>\!/Groups = !\

.ASCII \AS!\Multicast timer = !UB seconds!\<0>

.LONG 17694907

.ADDRESS P.AFH

.EXTRN SYSS\$FAOL

.PSECT \$CODE\$,NOWRT,2

.ENTRY LCP SET, Save R2,R3,R4

MOVAB LIB\$SIGNAL, R4

MOVAB CLIS\$PRESENT, R3

MOVAB -616(SP), SP

CALLS #0, LCP QUAL

MOVL R0, STATUS

BLBS STATUS, 1\$

MOVL STATUS, R0

RET

CALLS #0, LCP_CVTGROUPS

MOVAB NAMEDSC, P

MOVAB SERV\$DSC, P+4

MOVAB NODE_ID\$DSC, P+8

MOVAB ID\$DSC, P+12

MOVAB GRP\$DSC, P+16

MOVL MCASTIMR, P+20

MOVZWL #512, DSC

MOVAB STR, DSC+4

PUSHL SP

PUSHAB DSC

PUSHAB DSC

PUSHAB P.AFG

CALLS #4, SYSS\$FAOL

PUSHAB QUAL LOG

CALLS #1, CLIS\$PRESENT

BLBC R0, 2\$

PUSHAB DSC

CALLS #1, LIB\$PUT_OUTPUT

CLRL -(SP)

PUSHAB LCPK SET

CALLS #2, SYSS\$CMKRN

MOVL R0, STATUS

BLBS STATUS, 3\$

PUSHL STATUS

CLRL -(SP)

PUSHL #LCP\$ NOTSET

CALLS #3, LIB\$SIGNAL

	54	00000000G	00	9E	00002
	53	00000000G	00	9E	00009
	5E	FD98	CE	9E	00010
FA05	CF		00	FB	00015
	52		50	DO	0001A
	04		52	E8	0001D
	50		52	DO	00020
			04	00023	
0000V	CF		00	FB	00024
	6E	0000'	CF	9E	00029
04	AE	0000'	CF	9E	0002E
08	AE	0000'	CF	9E	00034
0C	AE	0000'	CF	9E	0003A
10	AE	0000'	CF	9E	00040
14	AE	0000'	CF	DO	00046
F8	AD	0200	8F	3C	0004C
FC	AD	60	AE	9E	00052
			5E	DD	00057
		F8	AD	9F	00059
		F8	AD	9F	0005C
		0000'	CF	9F	0005F
00000000G	00		04	FB	00063
		0000'	CF	9F	0006A
	63		01	FB	0006E
	0A		50	E9	00071
		F8	AD	9F	00074
00000000G	00		01	FB	00077
			7E	D4	0007E
		0000V	CF	9F	00080
00000000G	00		02	FB	00084
	52		50	DO	0008B
	0D		52	E8	0008E
			52	DD	00091
			7E	D4	00093
		00000000G	8F	DD	00095
	64		03	FB	0009B

2\$:

1613

1661

1663

1669

1675

1676

1677

1678

1679

1680

1682

1699

1701

1707

1709

LATCP
V04-000

LAT Control Program
LCP_SET Set multicast information

B 9
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 54
(11)

63	0000'	CF	9F	0009E	3\$:	PUSHAB	QUAL LOG	:	1715
09		01	FB	000A2		CALLS	#1, CLIPRESENT	:	
		50	E9	000A5		BLBC	R0, 4\$	:	
64	00000000G	8F	DD	000A8		PUSHL	#LCP\$ SET	:	
50		01	FB	000AE		CALLS	#1, LIB\$ SIGNAL	:	
		01	D0	000B1	4\$:	MOVL	#1, R0	:	1719
		04	000B4			RET		:	

; Routine Size: 181 bytes, Routine Base: \$CODE\$ + 06D1

LA
V0

```
: 1594 1720 1 %SBTTL 'LCP_SHOW Process a SHOW command'
: 1595 1721 1 GLOBAL ROUTINE LCP_SHOW : NOVALUE =
: 1596 1722 1
: 1597 1723 1 ++
: 1598 1724 1 FUNCTIONAL DESCRIPTION:
: 1599 1725 1
: 1600 1726 1 Routine to process an LCP SHOW ... command
: 1601 1727 1
: 1602 1728 1 FORMAL PARAMETERS:
: 1603 1729 1
: 1604 1730 1 NONE
: 1605 1731 1
: 1606 1732 1 IMPLICIT INPUTS:
: 1607 1733 1
: 1608 1734 1 CLI parsed command
: 1609 1735 1
: 1610 1736 1 IMPLICIT OUTPUTS:
: 1611 1737 1
: 1612 1738 1 NONE
: 1613 1739 1
: 1614 1740 1 ROUTINE VALUE:
: 1615 1741 1 COMPLETION CODES:
: 1616 1742 1
: 1617 1743 1 NONE
: 1618 1744 1
: 1619 1745 1 SIDE EFFECTS:
: 1620 1746 1
: 1621 1747 1 NONE
: 1622 1748 1
: 1623 1749 1 --
: 1624 1750 1
: 1625 1751 2 BEGIN
: 1626 1752 2
: 1627 1753 2 LOCAL
: 1628 1754 2 STATUS
: 1629 1755 2 ;
: 1630 1756 2
: 1631 1757 2 Obtain the subcommand
: 1632 1758 2
: 1633 1759 2 $SETDSC (SUBCMD DSC, CMDLEN, SUBCMD);
: 1634 1760 2 STATUS = CL$GET VALUE (QUAL_SUBCOMMAND, SUBCMD DSC);
: 1635 1761 2 IF .STATUS EQL C[IS_ABSENT
: 1636 1762 2 THEN
: 1637 1763 2 SUBCMD DSC [DSC$W_LENGTH] = 0
: 1638 1764 2
: 1639 1765 2 STR$TRIM (SUBCMD DSC, SUBCMD DSC, SUBCMD DSC [DSC$W_LENGTH] );
: 1640 1766 2
: 1641 1767 2 Dispatch on the subcommand string
: 1642 1768 2
: 1643 1769 2 SELECT ONE TRUE OF
: 1644 1770 2 SET
: 1645 1771 2 [ $CMPDSC (SUBCMD DSC, QUAL_COUNTERS) ] :
: 1646 1772 2 (
: 1647 1773 2 LCP_SHOW COUQUAL ();
: 1648 1774 2 LCP_SHOW COU ();
: 1649 1775 2 );
: 1650 1776 2
```


LATCP
V04-000

LAT Control Program
LCP_SHOW Process a SHOW command

D 9
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1 Page 56
(12)

```
: 1651      1777 2    [ $CMPDSC (SUBCMDDSC, QUAL_HISTORY) ] :
: 1652      1778      (
: 1653      1779      LCP_SHOWHIST ()
: 1654      1780      );
: 1655      1781
: 1656      1782 [ $CMPDSC (SUBCMDDSC, QUAL_CHAR) ] :
: 1657      1783      (
: 1658      1784      LCP_SHOWCHAR ()
: 1659      1785      );
: 1660      1786
: 1661      1787 [ $CMPDSC (SUBCMDDSC, QUAL_SERVERS) ] :
: 1662      1788      (
: 1663      1789      LCP_SHOWSERVERS ()
: 1664      1790      );
: 1665      1791
: 1666      1792 [OTHERWISE] : SIGNAL (LCP$_IVCMD);
: 1667      1793
: 1668      1794 TES
: 1669      1795
: 1670      1796 1    END;
```

				01FC 00000	.ENTRY	LCP_SHOW, Save R2,R3,R4,R5,R6,R7,R8	1721
	58	0000'	CF	9E 00002	MOVAB	SUBCMDDSC, R8	
	57	0000'	CF	9E 00007	MOVAB	QUAL_COUNTERS, R7	
	68		10	D0 0000C	MOVL	#16, SUBCMDDSC	1759
04	A8	F0	A8	9E 0000F	MOVAB	SUBCMD, SUBCMDDSC+4	
			58	DD 00014	PUSHL	R8	1760
		FF30	C7	9F 00016	PUSHAB	QUAL SUBCOMMAND	
00000000G	00		02	FB 0001A	CALLS	#2, CLISGET VALUE	
00000000G	8F		50	D1 00021	CMPL	STATUS, #CLIS_ABSENT	1761
			02	12 00028	BNEQ	1\$	
			68	B4 0002A	CLRW	SUBCMDDSC	1763
			58	DD 0002C	PUSHL	R8	1765
			58	DD 0002E	PUSHL	R8	
			58	DD 00030	PUSHL	R8	
00000000G	00		03	FB 00032	CALLS	#3, STR\$TRIM	
	55		68	3C 00039	MOVZWL	SUBCMDDSC, R5	1771
	56	04	A8	D0 0003C	MOVL	SUBCMDDSC+4, R6	
	50		55	D0 00040	MOVL	R5, R0	
	50		67	B1 00043	CMPL	QUAL_COUNTERS, R0	
			03	1E 00046	BGEQU	2\$	
	50		67	3C 00048	MOVZWL	QUAL_COUNTERS, R0	
	03		50	B1 0004B	CMPL	R0, #3	
			03	1E 0004E	BGEQU	3\$	
	50		03	D0 00050	MOVL	#3, R0	
50			54	D4 00053	CLRL	R4	
	20	66	55	2D 00055	CMPC5	R5, (R6), #32, R0, @QUAL_COUNTERS+4	
			04	B7 0005A			
			02	12 0005C	BNEQ	4\$	
			54	D6 0005E	INCL	R4	
	01		54	D1 00060	CMPL	R4, #1	
			0B	12 00063	BNEQ	5\$	
0000V	CF		00	FB 00065	CALLS	#0, LCP_SHOWCOUQUAL	1773

0000V	CF	00	FB	0006A	CALLS	#0, LCP_SHOWCOU	1774
		04	04	0006F	RET		
50		55	D0	00070	5\$:	MOVL R5, R0	1777
50	10	A7	B1	00073		CMPW QUAL_HISTORY, R0	
		04	1E	00077		BGEQU 6\$	
50	10	A7	3C	00079		MOVZWL QUAL_HISTORY, R0	
03		50	B1	0007D	6\$:	CMPW R0, #3	
		03	1E	00080		BGEQU 7\$	
50		03	D0	00082		MOVL #3, R0	
		54	D4	00085	7\$:	CLRL R4	
50	20	66	55	2D	00087	CMPC5 R5, (R6), #32, R0, @QUAL_HISTORY+4	
		14	B7	0008C			
		02	12	0008E		BNEQ 8\$	
		54	D6	00090		INCL R4	
01		54	D1	00092	8\$:	CMPL R4, #1	
		06	12	00095		BNEQ 9\$	
0000V	CF	00	FB	00097	CALLS	#0, LCP_SHOWHIST	1779
		04	04	0009C	RET		1778
50		55	D0	0009D	9\$:	MOVL R5, R0	1782
50	28	A7	B1	000A0		CMPW QUAL_CHAR, R0	
		04	1E	000A4		BGEQU 10\$	
50	28	A7	3C	000A6		MOVZWL QUAL_CHAR, R0	
03		50	B1	000AA	10\$:	CMPW R0, #3	
		03	1E	000AD		BGEQU 11\$	
50		03	D0	000AF		MOVL #3, R0	
		54	D4	000B2	11\$:	CLRL R4	
50	20	66	55	2D	000B4	CMPC5 R5, (R6), #32, R0, @QUAL_CHAR+4	
		2C	B7	000B9			
		02	12	000BB		BNEQ 12\$	
		54	D6	000BD		INCL R4	
01		54	D1	000BF	12\$:	CMPL R4, #1	
		06	12	000C2		BNEQ 13\$	
0000V	CF	00	FB	000C4	CALLS	#0, LCP_SHOWCHAR	1784
		04	04	000C9	RET		1783
50		55	D0	000CA	13\$:	MOVL R5, R0	1787
50	50	A7	B1	000CD		CMPW QUAL_SERVERS, R0	
		04	1E	000D1		BGEQU 14\$	
50	50	A7	3C	000D3		MOVZWL QUAL_SERVERS, R0	
03		50	B1	000D7	14\$:	CMPW R0, #3	
		03	1E	000DA		BGEQU 15\$	
50		03	D0	000DC		MOVL #3, R0	
		54	D4	000DF	15\$:	CLRL R4	
50	20	66	55	2D	000E1	CMPC5 R5, (R6), #32, R0, @QUAL_SERVERS+4	
		54	B7	000E6			
		02	12	000E8		BNEQ 16\$	
		54	D6	000EA		INCL R4	
01		54	D1	000EC	16\$:	CMPL R4, #1	
		06	12	000EF		BNEQ 17\$	
0000V	CF	00	FB	000F1	CALLS	#0, LCP_SHOWSERVERS	1789
		04	04	000F6	RET		1788
00000000G	00	00000000G	8F	DD	000F7	17\$:	1792
		01	FB	000FD	PUSHL	#LCP\$_IVCMD	
		04	04	0010A	CALLS	#1, LIB\$SIGNAL	
					RET		1796

; Routine Size: 261 bytes, Routine Base: \$CODE\$ + 0786

LATCP
V04-000

LAT Control Program
LCP_EXIT Process EXIT LCP request

F 9
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1 Page 58
(13)

```
: 1672 1797 1 %SBTTL 'LCP_EXIT Process EXIT LCP request'
: 1673 1798 1 GLOBAL ROUTINE LCP_EXIT : NOVALUE =
: 1674 1799 1
: 1675 1800 1 !++
: 1676 1801 1 FUNCTIONAL DESCRIPTION:
: 1677 1802 1
: 1678 1803 1 Routine to parse an LCP command line.
: 1679 1804 1
: 1680 1805 1 FORMAL PARAMETERS:
: 1681 1806 1
: 1682 1807 1 NONE
: 1683 1808 1
: 1684 1809 1 IMPLICIT INPUTS:
: 1685 1810 1
: 1686 1811 1 CLI parsed command
: 1687 1812 1
: 1688 1813 1 IMPLICIT OUTPUTS:
: 1689 1814 1
: 1690 1815 1 NONE
: 1691 1816 1
: 1692 1817 1 ROUTINE VALUE:
: 1693 1818 1 COMPLETION CODES:
: 1694 1819 1
: 1695 1820 1 NONE
: 1696 1821 1
: 1697 1822 1 SIDE EFFECTS:
: 1698 1823 1
: 1699 1824 1 NONE
: 1700 1825 1
: 1701 1826 1 --
: 1702 1827 1
: 1703 1828 2 BEGIN
: 1704 1829 2
: 1705 1830 2 LOCAL
: 1706 1831 2 STATUS
: 1707 1832 2 ;
: 1708 1833 2
: 1709 1834 3 STATUS = $EXIT (CODE = SS$_NORMAL)
: 1710 1835 3
: 1711 1836 1 END;
```

```
0000 00000
01 DD 00002
00000000G 00 01 FB 00004
04 0000B
```

.EXTRN SYS\$EXIT

```
.ENTRY LCP_EXIT, Save nothing
PUSHL #1
CALLS #1, SYS$EXIT
RET
```

```
: 1798
: 1834
: 1836
```

; Routine Size: 12 bytes, Routine Base: \$CODE\$ + 088B

```
1713 1837 1 %SBTTL 'LCPK_SET Kernel mode routine to set multicast data'
1714 1838 1 ROUTINE LCPK_SET =
1715 1839 1
1716 1840 1 ++
1717 1841 1 FUNCTIONAL DESCRIPTION:
1718 1842 1
1719 1843 1 Kernel mode routine to set the information in the buffer of the driver.
1720 1844 1
1721 1845 1 FORMAL PARAMETERS:
1722 1846 1
1723 1847 1 NONE
1724 1848 1
1725 1849 1 IMPLICIT INPUTS:
1726 1850 1
1727 1851 1 results of the qualifier parse
1728 1852 1
1729 1853 1 IMPLICIT OUTPUTS:
1730 1854 1
1731 1855 1 NONE
1732 1856 1
1733 1857 1 ROUTINE VALUE:
1734 1858 1 COMPLETION CODES:
1735 1859 1
1736 1860 1 NONE
1737 1861 1
1738 1862 1 SIDE EFFECTS:
1739 1863 1
1740 1864 1 qualifier results set for multicast
1741 1865 1
1742 1866 1 --
1743 1867 1
1744 1868 2 BEGIN
1745 1869 2
1746 1870 2 LOCAL
1747 1871 2 ZERO,
1748 1872 2 ONE,
1749 1873 2 TWO,
1750 1874 2 THIRTY_TWO,
1751 1875 2 LT : REF BLOCK [,BYTE],
1752 1876 2 PTR,
1753 1877 2 STR : REF BLOCK [,BYTE]
1754 1878 2 ;
1755 1879 2
1756 1880 2 ZERO = 0; ONE = 1; TWO = 2; THIRTY_TWO = 32;
1757 1881 2
1758 1882 2 Address of the buffer in the driver
1759 1883 2
1760 1884 2 LT = .LCPSL_AREA;
1761 1885 2 PTR = STR = .LT [GHBSL_NODE];
1762 1886 2
1763 1887 2 Copy the multicast timer, host node status and group codes
1764 1888 2
1765 1889 2 PTR = CH$MOVE (2, MCASTIMR, .PTR); ! Set timer and no host node status
1766 1890 2 PTR = CH$MOVE (1, THIRTY_TWO, .PTR); ! Thirty-Two bytes of group code
1767 1891 2 PTR = CH$MOVE (32, GROUPS, .PTR); ! Copy group codes
1768 1892 2
1769 1893 2 ! Copy the host node name and descriptor here
```



```
: 1770      1894      2      |
: 1771      1895      2      | PTR = CH$MOVE (1, NAMEDSC [DSC$W_LENGTH], .PTR);
: 1772      1896      2      | PTR = CH$MOVE (.NAMEDSC [DSC$W_LENGTH],
: 1773      1897      2      |      NAMEDSC [DSC$A_POINTER], .PTR);
: 1774      1898      2      | PTR = CH$MOVE (1, NODE_IDDSC [DSC$W_LENGTH], .PTR);
: 1775      1899      2      | PTR = CH$MOVE (.NODE_IDDSC [DSC$W_LENGTH],
: 1776      1900      2      |      .NODE_IDDSC [DSC$A_POINTER], .PTR);
: 1777      1901      2      |
: 1778      1902      2      | : If we have an service name, then report 2 service names here
: 1779      1903      2      |
: 1780      1904      2      | IF .SERVDSC [DSC$W_LENGTH] NEQ 0
: 1781      1905      2      | THEN
: 1782      1906      2      |     PTR = CH$MOVE (1, TWO, .PTR)
: 1783      1907      2      | ELSE
: 1784      1908      2      |     PTR = CH$MOVE (1, ONE, .PTR)
: 1785      1909      2      |
: 1786      1910      2      |
: 1787      1911      2      | :
: 1788      1912      2      | : Copy the first service rating, name and ID here
: 1789      1913      2      | : We will use the host node name as the first service name
: 1790      1914      2      |
: 1791      1915      2      | PTR = CH$MOVE (1, RATING, .PTR);
: 1792      1916      2      | PTR = CH$MOVE (1, NAMEDSC [DSC$W_LENGTH], .PTR);
: 1793      1917      2      | PTR = CH$MOVE (.NAMEDSC [DSC$W_LENGTH],
: 1794      1918      2      |      NAMEDSC [DSC$A_POINTER], .PTR);
: 1795      1919      2      | PTR = CH$MOVE (1, IDDSC [DSC$W_LENGTH], .PTR);
: 1796      1920      2      | PTR = CH$MOVE (.IDDSC [DSC$W_LENGTH], .IDDSC [DSC$A_POINTER], .PTR);
: 1797      1921      2      |
: 1798      1922      2      | :
: 1799      1923      2      | : If there is an service name, then build it and its rating as the
: 1800      1924      2      | : second service name
: 1801      1925      2      |
: 1802      1926      2      | IF .SERVDSC [DSC$W_LENGTH] NEQ 0
: 1803      1927      2      | THEN
: 1804      1928      2      |     BEGIN
: 1805      1929      2      |     PTR = CH$MOVE (1, SERVVRATING, .PTR);
: 1806      1930      2      |     PTR = CH$MOVE (1, SERVDSC [DSC$W_LENGTH], .PTR);
: 1807      1931      2      |     PTR = CH$MOVE (.SERVDSC [DSC$W_LENGTH],
: 1808      1932      2      |     .SERVDSC [DSC$A_POINTER], .PTR);
: 1809      1933      2      |     PTR = CH$MOVE (1, ZERO, .PTR) ! No ID for the second service name!
: 1810      1934      2      |     END
: 1811      1935      2      |
: 1812      1936      2      | :
: 1813      1937      2      | :
: 1814      1938      2      | : Now build the service classes
: 1815      1939      2      |
: 1816      1940      2      | PTR = CH$MOVE (1, ONE, .PTR);      ! One byte of service class
: 1817      1941      2      | PTR = CH$MOVE (1, ONE, .PTR);      ! Enable service class 0
: 1818      1942      2      |
: 1819      1943      2      | :
: 1820      1944      2      | : Store the total size of the data
: 1821      1945      2      |
: 1822      1946      2      | STR [-2, 0, 16, 0] = .PTR - .STR;
: 1823      1947      2      |
: 1824      1948      2      | :
: 1825      1949      2      | : Call the set entry of the driver
: 1826      1950      2      |
```

```
: 1827      1951 2      STARTSTOP (.LT [GHB$L_SETENTRY]);
: 1828      1952 2
: 1829      1953 2      Ignore the return from the driver for now.  It doesn't mean anything
: 1830      1954 2
: 1831      1955 2      SUCCESS
: 1832      1956 2
: 1833      1957 1      END;
```

```
OFFC 00000 LCPK_SET:
                                .WORD      Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11
                                CLRL      ZERO
                                MOVL      #1, ONE
                                MOVL      #2, TWO
                                MOVL      #32, THIRTY_TWO
                                MOVL      LCP$L_AREA, -LT
                                MOVL      28(LT), STR
                                MOVL      STR, PTR
                                MOVW      MCASTIMR, (PTR)+
                                MOVB      THIRTY_TWO, (PTR)+
                                MOVC3     #32, GROUPS, (PTR)
                                MOVB      NAMEDSC, (PTR)+
                                MOVC3     NAMEDSC, @NAMEDSC+4, (PTR)
                                MOVB      NODE_IDDSC, (PTR)+
                                MOVC3     NODE_IDDSC, @NODE_IDDSC+4, (PTR)
                                MOVZWL    SERVDSR, R8
                                CLRL      R9
                                TSTL      R8
                                BEQL      1$
                                INCL      R9
                                MOVB      TWO, (PTR)
                                BRB       2$
                                MOVB      ONE, (PTR)
                                INCL      PTR
                                MOVB      RATING, (PTR)+
                                MOVB      NAMEDSC, (PTR)+
                                MOVC3     NAMEDSC, @NAMEDSC+4, (PTR)
                                MOVB      IDDSC, (PTR)+
                                MOVC3     IDDSC, @IDDSC+4, (PTR)
                                BLBC      R9, 3$
                                MOVB      SERVDRATING, (PTR)+
                                MOVB      SERVDSR, (PTR)+
                                MOVC3     R8, @SERVDSR+4, (PTR)
                                MOVB      ZERO, (PTR)+
                                MOVB      ONE, (PTR)+
                                MOVB      ONE, (PTR)+
                                SUBW3     STR, PTR, -2(STR)
                                JSB       @4(LT)
                                MOVL      #1, R0
                                RET

5B      0000' 7E D4 00002
5A      0000' 01 D0 00004
50      0000' 02 D0 00007
57      0000' 20 D0 0000A
56      0000' CF D0 0000D
53      0000' A7 D0 00012
83      0000' 56 D0 00016
83      0000' CF B0 00019
63      0000' 50 90 0001E
83      0000' 20 28 00021
63      0000' CF 90 00027
83      0000' CF 28 0002C
63      0000' CF 90 00034
58      0000' CF 28 00039
58      0000' CF 3C 00041
59      0000' D4 00046
58      0000' D5 00048
07      0000' 13 0004A
59      0000' D6 0004C
63      0000' 5A 90 0004E
03      0000' 11 00051
63      0000' 5B 90 00053 1$:
53      0000' D6 00056 2$:
83      0000' CF 90 00058
83      0000' CF 90 0005D
63      0000' DF 28 00062
83      0000' CF 90 0006A
63      0000' DF 28 0006F
13      0000' 59 E9 00077
83      0000' CF 90 0007A
83      0000' CF 90 0007F
63      0000' DF 58 28 00084
83      0000' 6E 90 0008A
83      0000' 5B 90 0008D 3$:
83      0000' 5B 90 00090
FE A6    53      0000' 56 A3 00093
50      0000' B7 16 00098
50      0000' 01 D0 0009B
50      0000' 04 0009E
```

; Routine Size: 159 bytes, Routine Base: \$CODE\$ + 0897


```
1835 1958 1 %SBTTL 'LCPK_START Kernel mode start routine'
1836 1959 1 ROUTINE LCPK_START =
1837 1960 1
1838 1961 1 ++
1839 1962 1 FUNCTIONAL DESCRIPTION:
1840 1963 1
1841 1964 1 This routine is called in kernel mode to start the driver.
1842 1965 1 First it checks to see if the terminal driver is already active
1843 1966 1 by looking at the Timer Active flag. If
1844 1967 1 the device is not active, then a channel is assigned to the
1845 1968 1 data link device (a list of possible device names is tried) and
1846 1969 1 the ucb for the data link device is passed to the terminal port
1847 1970 1 driver.
1848 1971 1
1849 1972 1 FORMAL PARAMETERS:
1850 1973 1
1851 1974 1 NONE
1852 1975 1
1853 1976 1 IMPLICIT INPUTS:
1854 1977 1
1855 1978 1 LCP$L_AREA
1856 1979 1
1857 1980 1 IMPLICIT OUTPUTS:
1858 1981 1
1859 1982 1 NONE
1860 1983 1
1861 1984 1 ROUTINE VALUE:
1862 1985 1 COMPLETION CODES:
1863 1986 1
1864 1987 1 success or failure code
1865 1988 1
1866 1989 1 SIDE EFFECTS:
1867 1990 1
1868 1991 1 NONE
1869 1992 1
1870 1993 1 --
1871 1994 1
1872 1995 2 BEGIN
1873 1996 2
1874 1997 2 LOCAL
1875 1998 2 LT : REF BLOCK [,BYTE],
1876 1999 2 IOSB : VECTOR [4, WORD],
1877 2000 2 CCB : REF BLOCK [,BYTE],
1878 2001 2 UCB : REF BLOCK [,BYTE],
1879 2002 2 ORB : REF BLOCK [,BYTE],
1880 2003 2 DEV,
1881 2004 2 STATUS,
1882 2005 2 DEVCHAN : WORD
1883 2006 2 ;
1884 2007 2
1885 2008 2 |
1886 2009 2 | If the timer active flag is present, then the port driver is active.
1887 2010 2 |
1888 2011 2 LT = .LCP$L_AREA;
1889 2012 2 IF .LT [GHBSL_TIM_ACT] NEQ 0
1890 2013 2 THEN
1891 2014 2 RETURN SS$_DEVACTIVE
```

```
1892 2015 2
1893 2016 2
1894 2017 2
1895 2018 2
1896 2019 2
1897 2020 2
1898 2021 2
1899 2022 2
1900 2023 2
1901 2024 2
1902 2025 2
1903 2026 2
1904 2027 2
1905 2028 2
1906 2029 2
1907 2030 2
1908 2031 2
1909 2032 2
1910 2033 2
1911 2034 2
1912 2035 2
1913 2036 2
1914 2037 2
1915 2038 2
1916 2039 2
1917 2040 2
1918 P 2041 2
1919 P 2042 2
1920 P 2043 2
1921 P 2044 2
1922 P 2045 2
1923 P 2046 2
1924 2047 2
1925 2048 2
1926 2049 2
1927 2050 2
1928 2051 2
1929 2052 2
1930 2053 2
1931 2054 2
1932 2055 2
1933 2056 2
1934 2057 2
1935 2058 2
1936 2059 2
1937 2060 2
1938 2061 2
1939 2062 2
1940 2063 2
1941 2064 2
1942 2065 2
1943 2066 2
1944 2067 2
1945 2068 2
1946 2069 2
1947 2070 2
1948 2071 2

:
:
: Try each of the device names in turn
DEV = DEVNAMES;
WHILE ..DEV NEQ 0
DO
    BEGIN
        STATUS = $ASSIGN (DEVNAM = ..DEV, CHAN = DEVCHAN);
        IF .STATUS
        THEN
            EXITLOOP
        :
        DEV = .DEV + 4
    END
:
: If we never did find a device, then return the error
IF NOT .STATUS
THEN
    RETURN .STATUS
:
: Startup our protocol on the datalink device
STATUS = $QIOW
(
    FUNC = IOS$ SETMODE OR IOSM_CTRL OR IOSM_STARTUP,
    CHAN = .DEVCHAN,
    IOSB = IOSB,
    P2 = SETPARMDSC
);
:
: Something bad happened, so deassign our channel and then
: return the error
IF NOT .STATUS
OR
NOT ( STATUS = .IOSB [0])
THEN
    BEGIN
        $DASSGN (CHAN = .DEVCHAN);
        RETURN .STATUS
    END
:
: Call a handy system routine to find the datalink ucb address
: by obtaining the channel control block
IOS$VERIFYCHAN (.DEVCHAN; CCB);
UCB = .CCB [CCB$L_UCB];
:
: Call the start entry point of the port driver
: and if successful, then fix the ucb so it can stay around
STATUS = STARTSTOP (.LT [GHB$L_STENTRY], .UCB);
```



```

: 1949      2072 2      IF .STATUS
: 1950      2073 2      THEN
: 1951      2074 2      BEGIN
: 1952      2075 2      UCB [UCB$L_PID] = 0;
: 1953      2076 2      ORB = .UCB[UCB$L_ORB];
: 1954      2077 2      ORB [ORB$L_OWNER] = %X'10004';
: 1955      2078 2      ORB [ORB$W_PROT] = %X'FFFO';
: 1956      2079 2      END
: 1957      2080 2      :
: 1958      2081 2      : Deassign our channel to the datalink ucb
: 1959      2082 2      $DASSGN (CHAN = .DEVCHAN);
: 1960      2083 2      RETURN .STATUS
: 1961      2084 2
: 1962      2085 2
: 1963      2086 2
: 1964      2087 1      END;
```

```

.EXTRN SYSS$ASSIGN, SYSS$QIOW
.EXTRN SYSS$DASSGN
```

```

                                OFFC 00000 LCPK_START:
                                .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11
5E      0000' 10 C2 00002      SUBL2 #16, SP
56      18 A6 D0 00005      MOVL LCP$L_AREA, LT
                                TSTL 24(LT)
                                BEQL 1$
50      02C4 8F 3C 0000F      MOVZWL #708, R0
                                RET
52      0000' CF 9E 00015 1$: MOVAB DEVNAMES, DEV
                                TSTL (DEV)
                                BEQL 3$
                                CLRQ -(SP)
                                PUSHAB DEVCHAN
                                PUSHL (DEV)
00000000G 00 04 FB 00025      CALLS #4, SYSS$ASSIGN
57      50 D0 0002C      MOVL R0, STATUS
08      57 E8 0002F      BLBS STATUS, 4$
52      04 C0 00032      ADDL2 #4, DEV
                                BRB 2$
6A      57 E9 00037 3$: BLBC STATUS, 6$
                                CLRQ -(SP)
                                CLRQ -(SP)
                                PUSHAB SETPARMDSC
                                CLRQ -(SP)
                                CLRL -(SP)
                                PUSHAB IOSB
7E      0263 8F 3C 00049      MOVZWL #611, -(SP)
7E      2C AE 3C 0004E      MOVZWL DEVCHAN, -(SP)
                                CLRL -(SP)
00000000G 00 0C FB 00054      CALLS #12, SYSS$QIOW
57      50 D0 0005B      MOVL R0, STATUS
38      57 E9 0005E      BLBC STATUS, 5$
57      08 AE 3C 00061      MOVZWL IOSB, STATUS
31      57 E9 00065      BLBC STATUS, 5$
50      04 AE 3C 00068      MOVZWL DEVCHAN, R0
```

LATCP
V04-000

LAT Control Program
LCPK_START Kernel mode start routine

M 9
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 65
(15)

		00000000G	00	16	0006C	JSB	IOC\$VERIFYCHAN	:	
	6E		61	D0	00072	MOVL	(CCB), UCB	:	2066
	55		6E	D0	00075	MOVL	UCB, R5	:	2071
		10	B6	16	00078	JSB	@16(LT)	:	
	57		50	D0	0007B	MOVL	R0, STATUS	:	
	18		57	E9	0007E	BLBC	STATUS, 5\$	:	2072
50	6E		2C	C1	00081	ADDL3	#44, UCB, R0	:	2075
			60	D4	00085	CLRL	(R0)	:	
51	6E		1C	C1	00087	ADDL3	#28, UCB, R1	:	2076
	50		61	D0	0008B	MOVL	(R1), ORB	:	
	60	00010004	8F	D0	0008E	MOVL	#65540, (ORB)	:	2077
	18		A0	AE	00095	MNEGW	#16, 24(ORB)	:	2078
	7E	04	AE	3C	00099	MOVZWL	DEVCHAN, -(SP)	:	2084
00000000G	00		01	FB	0009D	CALLS	#1, SYS\$DASSGN	:	
	50		57	D0	000A4	MOVL	STATUS, R0	:	2085
			04	000A7	6\$:	RET		:	2087

; Routine Size: 168 bytes, Routine Base: \$CODE\$ + 0936


```
: 1966      2088 1 %SBTTL 'LCPK_STOP' Kernel routine to stop the port driver'
: 1967      2089 1 ROUTINE LCPK_STOP =
: 1968      2090 1
: 1969      2091 1 ++
: 1970      2092 1 FUNCTIONAL DESCRIPTION:
: 1971      2093 1
: 1972      2094 1 This routine is called in kernel mode and calls the shut entry point
: 1973      2095 1 in the ldriver to stop all circuit and run things down.
: 1974      2096 1
: 1975      2097 1 FORMAL PARAMETERS:
: 1976      2098 1
: 1977      2099 1 NONE
: 1978      2100 1
: 1979      2101 1 IMPLICIT INPUTS:
: 1980      2102 1
: 1981      2103 1 LCP$L_AREA
: 1982      2104 1
: 1983      2105 1 IMPLICIT OUTPUTS:
: 1984      2106 1
: 1985      2107 1 NONE
: 1986      2108 1
: 1987      2109 1 ROUTINE VALUE:
: 1988      2110 1 COMPLETION CODES:
: 1989      2111 1
: 1990      2112 1 success or failure code
: 1991      2113 1
: 1992      2114 1 SIDE EFFECTS:
: 1993      2115 1
: 1994      2116 1 NONE
: 1995      2117 1
: 1996      2118 1 --
: 1997      2119 1
: 1998      2120 2 BEGIN
: 1999      2121 2
: 2000      2122 2 LITERAL DEVNAMSIZE = 16;
: 2001      2123 2
: 2002      2124 2 BIND
: 2003      2125 2 DEVNAMCTR = DESCRIPTOR ('!AC!UW')
: 2004      2126 2 ;
: 2005      2127 2
: 2006      2128 2 LOCAL
: 2007      2129 2 LT : REF BLOCK [,BYTE],
: 2008      2130 2 DEVNAMSTR : VECTOR [DEVNAMSIZE, BYTE],
: 2009      2131 2 DEVNAMDSC : BLOCK [DSC$K_Z_BLN, BYTE],
: 2010      2132 2 UCB : REF BLOCK [,BYTE],
: 2011      2133 2 ORB : REF BLOCK [,BYTE],
: 2012      2134 2 DDB : REF BLOCK [,BYTE],
: 2013      2135 2 IOSB : VECTOR [4, WORD],
: 2014      2136 2 STATUS,
: 2015      2137 2 DEVCHAN : WORD
: 2016      2138 2 ;
: 2017      2139 2
: 2018      2140 2 !
: 2019      2141 2 ! If the data link UCB is gone, then we are inactive
: 2020      2142 2
: 2021      2143 2 LT = .LCP$L_AREA;
: 2022      2144 2 IF (UCB = .[T [GHB$L_UCB] ) EQL 0
```

```
2023 2145 2 THEN
2024 2146 2     RETURN SSS_DEVINACT
2025 2147 2
2026 2148 2
2027 2149 2     Get pointers to the ucb and ddb for datalink to make its name
2028 2150 2     so we can assign a channel to the data link device ucb
2029 2151 2
2030 2152 2 DDB = .UCB [UCB$L_DDB];
2031 2153 2 DEVNAMDSC [DSC$A_POINTER] = DEVNAMSTR;
2032 2154 2 DEVNAMDSC [DSC$W_LENGTH] = DEVNAMSIZE;
2033 2155 2 STATUS = $FAO
2034 2156 2 (
2035 2157 2     DEVNAMCTR,
2036 2158 2     DEVNAMDSC [DSC$W_LENGTH],
2037 2159 2     DEVNAMDSC,
2038 2160 2     DDB [DDB$I_NAME],
2039 2161 2     .UCB [UCB$W_UNIT]
2040 2162 2 );
2041 2163 2 IF NOT .STATUS
2042 2164 2 THEN
2043 2165 2     RETURN .STATUS
2044 2166 2
2045 2167 2
2046 2168 2     Fix the ucb so we can assign a channel to it.
2047 2169 2
2048 2170 2 UCB [UCB$L_PID] = 0;
2049 2171 2 ORB = .UCB [UCB$L_ORB];
2050 2172 2 ORB [ORB$L_OWNER] = 0;
2051 2173 2 ORB [ORB$W_PROT] = 0;
2052 2174 2
2053 2175 2     Assign a channel to hold it here and allow us to shut it down
2054 2176 2     later and deassign the ucb to cause it to go away.
2055 2177 2
2056 2178 2 STATUS = $ASSIGN (DEVNAM = DEVNAMDSC, CHAN = DEVCHAN);
2057 2179 2 IF NOT .STATUS
2058 2180 2 THEN
2059 2181 2     RETURN .STATUS
2060 2182 2
2061 2183 2
2062 2184 2     Call the shutdown entry of the terminal port driver
2063 2185 2
2064 2186 2 STARTSTOP (.LT [GHB$L_SHUTENTRY]);
2065 2187 2
2066 2188 2     Shut down the datalink unit to free the protocol type
2067 2189 2
2068 2190 2 STATUS = $QIOW
2069 2191 2 (
2070 2192 2     FUNC = IOS_SETMODE OR IOSM_CTRL OR IOSM_SHUTDOWN,
2071 2193 2     CHAN = .DEVCHAN,
2072 2194 2     IOSB = IOSB
2073 2195 2 );
2074 2196 2
2075 2197 2     Collect any failures in status
2076 2198 2
2077 2199 2 IF NOT .STATUS OR NOT (STATUS = .IOSB [0])
2078 2200 2 THEN
2079 2201 2     0
```



```

: 2080      2202  2      ;
: 2081      2203  2      ;
: 2082      2204  2      ;
: 2083      2205  2      ; Temporary code to clean up io queue for mailbox delete
: 2084      2206  2      ;
: 2085      2207  2      ; UCB [UCB$L_IOBL] = UCB [UCB$L_IOFL] = 0;
: 2086      2208  2      ;
: 2087      2209  2      ;
: 2088      2210  2      ; Deassign the channel that we have. If shutentry succeeded, then
: 2089      2211  2      ; the device will disappear
: 2090      2212  2      ;
: 2091      2213  2      ; $DASSGN (CHAN = .DEVCHAN);
: 2092      2214  2      ;
: 2093      2215  2      ; RETURN .STATUS
: 2094      2216  2      ;
: 2095      2217  1      ; END;
```

```

.PSECT $PLITS$,NOWRT,NOEXE,2
57 55 21 43 41 21 00DD4 P.AFJ: .ASCII \!AC!UW\
                                00DDA .BLKB 2
                                00000006 00DDC P.AFI: .LONG 6
                                00000000 00DE0 .ADDRESS P.AFJ
                                ;
                                ;
                                ;
DEVNAMCTR= P.AFI
.EXTRN SYSSFAO
.PSECT $CODE$,NOWRT,2
OFFC 00000 LCPK_STOP:
5E      28 C2 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11 : 2089
53      0000' CF D0 00005 .SUBL2 #40, SP
52      14 A3 D0 0000A .MOVL LCP$L_AREA, LT : 2143
        06 12 0000E .MOVL 20(LT), UCB : 2144
50      20D4 8F 3C 00010 .BNEQ 1$
        04 00015 .MOVZWL #8404, R0 : 2146
50      28 A2 D0 00016 1$: .RET
14      AE 18 AE 9E 0001A .MOVL 40(UCB), DDB : 2152
10      AE 10 B0 0001F .MOVAB DEVNAMSTR, DEVNAMDSC+4 : 2153
        7E 54 A2 3C 00023 .MOVW #16, DEVNAMDSC : 2154
        14 A0 9F 00027 .MOVZWL 84(UCB), -(SP) : 2162
        18 AE 9F 0002A .PUSHAB 20(DDb)
        1C AE 9F 0002D .PUSHAB DEVNAMDSC
        0000' CF 9F 00030 .PUSHAB DEVNAMDSC
00000000G 00 05 FB 00034 .PUSHAB DEVNAMCTR
        6E 50 D0 0003B .CALLS #5, SYSSFAO
        56 6E E9 0003E .MOVL R0, STATUS
        2C A2 D4 00041 .BLBC STATUS, 3$ : 2163
        1C A2 D0 00044 .CLRL 44(UCB) : 2170
        60 D4 00048 .MOVL 28(UCB), ORB : 2171
        18 A0 B4 0004A .CLRL (ORB) : 2172
        7E 7C 0004D .CLRW 24(ORB) : 2173
        0C AE 9F 0004F .CLRQ -(SP) : 2178
        1C AE 9F 00052 .PUSHAB DEVCHAN
        .PUSHAB DEVNAMDSC
                                ;
```

LATCP
V04-000

LAT Control Program
LCPK_STOP Kernel routine to stop the port driv

D 10
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 69
(16)

00000000G	00	04	FB	00055	CALLS	#4, SYSS\$ASSIGN	:	
	6E	50	D0	0005C	MOVL	R0, STATUS	:	
	35	6E	E9	0005F	BLBC	STATUS, 3\$	:	2179
		0C	B3	16	JSB	@12(LT)	:	2186
			7E	7C	CLRQ	-(SP)	:	2195
			7E	7C	CLRQ	-(SP)	:	
			7E	7C	CLRQ	-(SP)	:	
			7E	7C	CLRQ	-(SP)	:	
		28	AE	9F	PUSHAB	IOSB	:	
	7E	02A3	8F	3C	MOVZWL	#675, -(SP)	:	
	7E	2C	AE	3C	MOVZWL	DEVCHAN, -(SP)	:	
			7E	D4	CLRL	-(SP)	:	
00000000G	00		0C	FB	CALLS	#12, SYSS\$QIOW	:	
	6E		50	D0	MOVL	R0, STATUS	:	
	04		6E	E9	BLBC	STATUS, 2\$	:	2199
	6E	08	AE	3C	MOVZWL	IOSB, STATUS	:	
	7E	04	AE	3C	MOVZWL	DEVCHAN, -(SP)	:	2213
00000000G	00		01	FB	CALLS	#1, SYSS\$DASSGN	:	
	50		6E	D0	MOVL	STATUS, R0	:	2215
			04	0009A	RET		:	2217

; Routine Size: 155 bytes, Routine Base: \$CODE\$ + 09DE


```
2097 2218 1 %SBTTL 'LCPK_STARTHISTORY Kernel mode start history routine'
2098 2219 1 ROUTINE LCPK_STARTHISTORY =
2099 2220 1
2100 2221 1 ++
2101 2222 1 FUNCTIONAL DESCRIPTION:
2102 2223 1
2103 2224 1 This routine is called in kernel mode to start history journalling
2104 2225 1 in the driver. First it checks to see if the terminal driver is
2105 2226 1 already active by looking at the Timer Active flag. If
2106 2227 1 the device is not active, then an error is returned, else the history
2107 2228 1 buffer is allocated and pointed to in the driver global area.
2108 2229 1
2109 2230 1 FORMAL PARAMETERS:
2110 2231 1
2111 2232 1 NONE
2112 2233 1
2113 2234 1 IMPLICIT INPUTS:
2114 2235 1
2115 2236 1 LCPSL_AREA
2116 2237 1
2117 2238 1 IMPLICIT OUTPUTS:
2118 2239 1
2119 2240 1 NONE
2120 2241 1
2121 2242 1 ROUTINE VALUE:
2122 2243 1 COMPLETION CODES:
2123 2244 1
2124 2245 1 success or failure code
2125 2246 1
2126 2247 1 SIDE EFFECTS:
2127 2248 1
2128 2249 1 NONE
2129 2250 1
2130 2251 1 --
2131 2252 1
2132 2253 2 BEGIN
2133 2254 2
2134 2255 2 LOCAL
2135 2256 2 LT : REF BLOCK [,BYTE],
2136 2257 2 HIST_BUF : REF BLOCK [,BYTE],
2137 2258 2 STATUS
2138 2259 2 ;
2139 2260 2
2140 2261 2
2141 2262 2 Check to make sure the driver is active.
2142 2263 2
2143 2264 2 LT = .LCPSL_AREA;
2144 2265 2 IF .LT [GHB$T_TIM_ACT] EQL 0
2145 2266 2 THEN
2146 2267 2 RETURN SS$_DEVINACT
2147 2268 2 ;
2148 2269 2
2149 2270 2
2150 2271 2 Try to allocate a buffer, else insufficient memory
2151 2272 2
2152 2273 3 IF NOT (STATUS = EX$_ALONONPAGED (GHB$_HISTSZ+HBF_C_LENGTH; HIST_BUF) )
2153 2274 2 THEN
```

; Routine Size: 78 bytes, Routine Base: \$CODES + 0A79


```
: 2172      2292 1 %SBTTL 'LCPK_STOPHISTORY Kernel mode stop history routine'
: 2173      2293 1 ROUTINE LCPK_STOPHISTORY : NOVALUE =
: 2174      2294 1
: 2175      2295 1 ++
: 2176      2296 1 FUNCTIONAL DESCRIPTION:
: 2177      2297 1
: 2178      2298 1 This routine is called in kernel mode to start history journalling
: 2179      2299 1 in the driver. First it checks to see if the terminal driver is
: 2180      2300 1 already active by looking at the Timer Active flag. If
: 2181      2301 1 the device is not active, then an error is returned, else the history
: 2182      2302 1 buffer is allocated and pointed to in the driver global area.
: 2183      2303 1
: 2184      2304 1 FORMAL PARAMETERS:
: 2185      2305 1
: 2186      2306 1 NONE
: 2187      2307 1
: 2188      2308 1 IMPLICIT INPUTS:
: 2189      2309 1
: 2190      2310 1 LCP$ _AREA
: 2191      2311 1
: 2192      2312 1 IMPLICIT OUTPUTS:
: 2193      2313 1
: 2194      2314 1 NONE
: 2195      2315 1
: 2196      2316 1 ROUTINE VALUE:
: 2197      2317 1 COMPLETION CODES:
: 2198      2318 1
: 2199      2319 1 NONE
: 2200      2320 1
: 2201      2321 1 SIDE EFFECTS:
: 2202      2322 1
: 2203      2323 1 NONE
: 2204      2324 1
: 2205      2325 1 --
: 2206      2326 1
: 2207      2327 2 BEGIN
: 2208      2328 2
: 2209      2329 2 LOCAL
: 2210      2330 2 HIST_BUF : REF BLOCK [,BYTE]
: 2211      2331 2 ;
: 2212      2332 2
: 2213      2333 2 |
: 2214      2334 2 Check to make sure the driver is active.
: 2215      2335 2
: 2216      2336 2 IF .LCP$ _AREA [GHBS$ _HISTORY] EQL 0
: 2217      2337 2 THEN
: 2218      2338 2 RETURN
: 2219      2339 2 ;
: 2220      2340 2
: 2221      2341 2 |
: 2222      2342 2 Get history buffer and deallocate it.
: 2223      2343 2
: 2224      2344 2 HIST_BUF = .LCP$ _AREA [GHBS$ _HISTORY];
: 2225      2345 2 LCP$[ _AREA [GHBS$ _HISTORY] = 0;
: 2226      2346 2 EXES$DEANONPAGED (HIST_BUF);
: 2227      2347 2
: 2228      2348 1 END;
```

```

                                OFFC 00000 LCPK_STOPHISTORY:
5E                                .WORD      Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11      : 2293
50                                #4, SP                                           :
                                0000' CF D0 00005                                MOVL    LCP$L_AREA, R0                      : 2336
                                08  A0 D5 0000A                                TSTL    8(R0)                            :
                                10  13 0000D                                BEQL    1$                               :
6E                                08  A0 D0 0000F                                MOVL    8(R0), HIST_BUF                    : 2344
                                08  A0 D4 00013                                CLRL    8(R0)                            : 2345
50                                6E  9E 00016                                MOVAB   HIST_BUF, R0                      : 2346
                                00000000G 00 15 00019                                JSB     EXE$DEANONPAGED                   :
                                04 0001F 1$:                                RET                                         : 2348

```

; Routine Size: 32 bytes, Routine Base: \$CODE\$ + 0AC7


```
2230 2349 1 %SBTTL 'LCPK_IS_LAT_TERMINAL Kernel routine to determine if TT is LAT terminal'
2231 2350 1 ROUTINE LCPK_IS_LAT_TERMINAL =
2232 2351 1
2233 2352 1 ++
2234 2353 1 FUNCTIONAL DESCRIPTION:
2235 2354 1
2236 2355 1 This routine is called in kernel mode to determine if the current
2237 2356 1 TT: device is a LAT terminal.
2238 2357 1
2239 2358 1 FORMAL PARAMETERS:
2240 2359 1
2241 2360 1 NONE
2242 2361 1
2243 2362 1 IMPLICIT INPUTS:
2244 2363 1
2245 2364 1 NONE
2246 2365 1
2247 2366 1 IMPLICIT OUTPUTS:
2248 2367 1
2249 2368 1 NONE
2250 2369 1
2251 2370 1 ROUTINE VALUE:
2252 2371 1 COMPLETION CODES:
2253 2372 1
2254 2373 1 success or failure code
2255 2374 1
2256 2375 1 SIDE EFFECTS:
2257 2376 1
2258 2377 1 NONE
2259 2378 1
2260 2379 1 --
2261 2380 1
2262 2381 2 BEGIN
2263 2382 2
2264 2383 2 BIND
2265 2384 2 TTNAME = DESCRIPTOR ('TT'),
2266 2385 2 DRVNAM = DESCRIPTOR ('LTDRIVER') : BLOCK [,BYTE]
2267 2386 2 ;
2268 2387 2
2269 2388 2 LOCAL
2270 2389 2 UCB : REF BLOCK [,BYTE],
2271 2390 2 CCB : REF BLOCK [,BYTE],
2272 2391 2 DDB : REF BLOCK [,BYTE],
2273 2392 2 STATUS,
2274 2393 2 DEVCHAN : WORD
2275 2394 2 ;
2276 2395 2
2277 2396 2 |
2278 2397 2 If we can't assign a channel, then return success, to prevent
2279 2398 2 shutting down LAT.
2280 2399 2
2281 2400 3 IF NOT (STATUS = $ASSIGN (DEVNAM = TTNAME, CHAN = DEVCHAN) )
2282 2401 2 THEN
2283 2402 2 RETURN SUCCESS
2284 2403 2 ;
2285 2404 2
2286 2405 2 |
```

```
: 2287      2406 2      | Call a handy system routine to find the terminal ucb address
: 2288      2407 2      | from the channel control block.
: 2289      2408 2      |
: 2290      2409 2      | IOC$VERIFYCHAN (.DEVCHAN; CCB);
: 2291      2410 2      | UCB = .CCB [CCB$$_UCB];          ! Get (virtual?) UCB address
: 2292      2411 2      |
: 2293      2412 2      | Deassign our channel to the terminal ucb
: 2294      2413 2      |
: 2295      2414 2      | $DASSGN (CHAN = .DEVCHAN);
: 2296      2415 2      |
: 2297      2416 2      | IF .UCB [UCB$$_TL_PHYUCB] EQL 0    ! LT devices should have this
: 2298      2417 2      | THEN                               ! field set!
: 2299      2418 2      |     RETURN FAILURE                ! Assume we're not an LT term
: 2300      2419 2      |
: 2301      2420 2      |
: 2302      2421 2      | UCB = .UCB [UCB$$_TL_PHYUCB];      ! Get Physical UCB address
: 2303      2422 2      | DDB = .UCB [UCB$$_DDB];           ! Get DDB address
: 2304      2423 2      |
: 2305      2424 2      |
: 2306      2425 2      | Return the result of the comparison
: 2307      2426 2      |
: 2308      2427 2      | CH$EQL
: 2309      2428 2      | (
: 2310      2429 2      |     .DRVNAM [DSC$_LENGTH], .DRVNAM [DSC$_POINTER],
: 2311      2430 2      |     CH$RCHAR (DDB [DDB$_DRVNAM_LEN]),
: 2312      2431 2      |     DDB [DDB$_DRVNAME] +1,
: 2313      2432 2      |     0
: 2314      2433 2      | )
: 2315      2434 2      |
: 2316      2435 1      | END;
```

```
.PSECT $SPLITS$,NOWRT,NOEXE,2

      54 54 00DE4 P.AFL: .ASCII \TT\
      00000002 00DE6 .BLKB 2
      00000000 00DE8 P.AFK: .LONG 2
      52 45 56 49 52 44 54 4C 00DF0 P.AFN: .ADDRESS P.AFL
      00000008 00DF8 P.AFM: .ASCII \LTDRIVER\
      00000000 00DFC P.AFM: .LONG 8
      .ADDRESS P.AFN

      TTNAME= P.AFK
      DRVNAM= P.AFM

.PSECT $CODE$,NOWRT,2

      OFFC 0000 LCPK_IS_LAT_TERMINAL:
      5E      04 C2 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11 : 2350
      7E 7C 00005 .SUBL2 #4, SP
      08 AE 9F 00007 .CLRQ -(SP) : 2400
      0000 CF 9F 0000A .PUSHAB DEVCHAN
      0000000G 00 04 FB 0000E .PUSHAB TTNAME
      50 E8 00015 .CALLS #4, SYSS$ASSIGN
      .BLBS STATUS, 1$
```


LATCP
V04-000

LAT Control Program
LCPK_IS_LAT_TERMINAL

Kernel routine to determi

K 10
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 76
(19)

50		01	D0	00018	MOVL	#1, R0	2402	
			04	0001B	RET			
50		6E	3C	0001C	1\$:	MOVZWL	DEVCHAN, R0	2409
	00000000G	00	16	0001F		JSB	IOCSVERIFYCHAN	
52		61	D0	00025		MOVL	(CCB), UCB	2410
7E		6E	3C	00028		MOVZWL	DEVCHAN, -(SP)	2414
	00000000G	00	01	FB	0002B	CALLS	#1, SYSSDASSGN	
50		C2	D0	00032		MOVL	160(UCB), R0	2416
	00A0	03	12	00037		BNEQ	2\$	
		50	D4	00039		CLRL	R0	2418
			04	0003B		RET		
52		50	D0	0003C	2\$:	MOVL	R0, UCB	2421
50		28	A2	D0	0003F	MOVL	40(UCB), DDB	2422
51		24	A0	9A	00043	MOVZBL	36(DDB), R1	2430
			54	D4	00047	CLRL	R4	
51	00	0000'	DF	0000'	CF	2D	00049	
				25	A0		00052	
					02	12	00054	
					54	D6	00056	
					54	D0	00058	3\$:
					04	0005B		
						BNEQ	3\$	
						INCL	R4	
						MOVL	R4, R0	2435
						RET		

; Routine Size: 92 bytes, Routine Base: \$CODE\$ + 0AE7

```
: 2318 2436 1 %SBTTL 'LCP ZERO Zero counters and history buffer'
: 2319 2437 1 GLOBAL ROUTINE LCP_ZERO : NOVALUE =
: 2320 2438 1
: 2321 2439 1 ++
: 2322 2440 1 FUNCTIONAL DESCRIPTION:
: 2323 2441 1
: 2324 2442 1 Call the kernel mode routine to zero the counters and history buffer.
: 2325 2443 1
: 2326 2444 1 FORMAL PARAMETERS:
: 2327 2445 1
: 2328 2446 1 NONE
: 2329 2447 1
: 2330 2448 1 IMPLICIT INPUTS:
: 2331 2449 1
: 2332 2450 1 NONE
: 2333 2451 1
: 2334 2452 1 IMPLICIT OUTPUTS:
: 2335 2453 1
: 2336 2454 1 NONE
: 2337 2455 1
: 2338 2456 1 ROUTINE VALUE:
: 2339 2457 1 COMPLETION CODES:
: 2340 2458 1
: 2341 2459 1 NONE
: 2342 2460 1
: 2343 2461 1 SIDE EFFECTS:
: 2344 2462 1
: 2345 2463 1 NONE
: 2346 2464 1
: 2347 2465 1 --
: 2348 2466 1
: 2349 2467 2 BEGIN
: 2350 2468 2
: 2351 2469 3 $CMKRNL (ROUTIN = LCPK_ZERO)
: 2352 2470 3
: 2353 2471 1 END;
```

```
0000 00000
7E D4 00002
0000V CF 9F 00004
00000000G 00 02 FB 00008
04 0000F
```

```
.ENTRY LCP_ZERO, Save nothing
CLRL -(SP)
PUSHAB LCPK_ZERO
CALLS #2, SYSSCMKRNL
RET
```

```
: 2437
: 2469
:
: 2471
```

; Routine Size: 16 bytes, Routine Base: \$CODE\$ + 0B43


```
: 2355      2472 1 %SBTTL 'LCPK_ZERO Kernel mode zero routine'
: 2356      2473 1 ROUTINE LCPK_ZERO : NOVALUE =
: 2357      2474 1
: 2358      2475 1 ++
: 2359      2476 1 FUNCTIONAL DESCRIPTION:
: 2360      2477 1
: 2361      2478 1 Zero the counter block and the history buffer
: 2362      2479 1
: 2363      2480 1 FORMAL PARAMETERS:
: 2364      2481 1
: 2365      2482 1 NONE
: 2366      2483 1
: 2367      2484 1 IMPLICIT INPUTS:
: 2368      2485 1
: 2369      2486 1 NONE
: 2370      2487 1
: 2371      2488 1 IMPLICIT OUTPUTS:
: 2372      2489 1
: 2373      2490 1 NONE
: 2374      2491 1
: 2375      2492 1 ROUTINE VALUE:
: 2376      2493 1 COMPLETION CODES:
: 2377      2494 1
: 2378      2495 1 NONE
: 2379      2496 1
: 2380      2497 1 SIDE EFFECTS:
: 2381      2498 1
: 2382      2499 1 NONE
: 2383      2500 1
: 2384      2501 1 --
: 2385      2502 1
: 2386      2503 2 BEGIN
: 2387      2504 2
: 2388      2505 2 LOCAL
: 2389      2506 2 LT : REF BLOCK [,BYTE],
: 2390      2507 2 HIST_BUF : REF BLOCK [,BYTE]
: 2391      2508 2 ;
: 2392      2509 2
: 2393      2510 2 LT = .LCP$SL_AREA;
: 2394      2511 2
: 2395      2512 2 Zero the counterblock
: 2396      2513 2
: 2397      2514 2 CH$FILL (0, GHBSK_CTRLLENGTH, LT [GHBSL_COUNTERS] );
: 2398      2515 2
: 2399      2516 2
: 2400      2517 2 Zero the history buffer if there is one
: 2401      2518 2
: 2402      2519 2 IF .LT [GHBSL_HISTORY] NEQ 0
: 2403      2520 2 THEN
: 2404      2521 3 BEGIN
: 2405      2522 3 HIST_BUF = .LT [GHBSL_HISTORY];
: 2406      2523 3 CH$FILL (0, GHBSK_HISTSIZE, HIST_BUF [HBF_Z_DATA]);
: 2407      2524 3 END
: 2408      2525 2 ;
: 2409      2526 2
: 2410      2527 1 END;
```

LATCP
V04-000

LAT Control Program
LCPK_ZERO Kernel mode zero routine

N 10
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 79
(21)

007C 00000 LCPK_ZERO:											
			56	0000'	CF	D0	00002	.WORD	Save R2,R3,R4,R5,R6	:	2473
			6E		00	2C	00007	MOVL	LCP\$ AREA, LT	:	2510
				2C	A6		0000C	MOVCS	#0, (SP), #0, #52, 44(LT)	:	2514
				08	A6	D5	0000E	TSTL	8(LT)	:	2519
					0D	13	00011	BEQL	1\$	:	
			50	08	A6	D0	00013	MOVL	8(LT), HIST_BUF	:	2522
0A00	8F	00	6E		00	2C	00017	MOVCS	#0, (SP), #0, #2560, 12(HIST_BUF)	:	2523
				0C	A0		0001E			:	
						04	00020	1\$:	RET	:	2527

; Routine Size: 33 bytes, Routine Base: \$CODE\$ + 0B53


```
: 2412 2528 1 %SBTTL 'LCP_SHOWCOUQUAL Process show qualifiers'
: 2413 2529 1 ROUTINE LCP_SHOWCOUQUAL : NOVALUE =
: 2414 2530 1
: 2415 2531 1 ++
: 2416 2532 1 FUNCTIONAL DESCRIPTION:
: 2417 2533 1
: 2418 2534 1 Process qualifiers for the SHOW COUNTERS command. Convert the qualifiers
: 2419 2535 1 to descriptors to set in the driver.
: 2420 2536 1
: 2421 2537 1 FORMAL PARAMETERS:
: 2422 2538 1
: 2423 2539 1 NONE
: 2424 2540 1
: 2425 2541 1 IMPLICIT INPUTS:
: 2426 2542 1
: 2427 2543 1 cli processed qualifiers
: 2428 2544 1
: 2429 2545 1 IMPLICIT OUTPUTS:
: 2430 2546 1
: 2431 2547 1 qualifier descriptors
: 2432 2548 1
: 2433 2549 1 ROUTINE VALUE:
: 2434 2550 1 COMPLETION CODES:
: 2435 2551 1
: 2436 2552 1 errors signalled
: 2437 2553 1
: 2438 2554 1 SIDE EFFECTS:
: 2439 2555 1
: 2440 2556 1 NONE
: 2441 2557 1
: 2442 2558 1 --
: 2443 2559 1
: 2444 2560 2 BEGIN
: 2445 2561 2
: 2446 2562 2 |
: 2447 2563 2 | All qualifiers are optional!
: 2448 2564 2 |
: 2449 2565 2 | FLAGS [FLAG_V_NODE] = CLISPRESNT (QUAL_NODE);
: 2450 2566 2 | FLAGS [FLAG_V_DEVICE] = CLISPRESNT (QUAL_DEVICE);
: 2451 2567 2 | FLAGS [FLAG_V_SERVERS] = CLISPRESNT (QUAL_SERVERS);
: 2452 2568 2 |
: 2453 2569 2 |
: 2454 2570 2 | If no qualifier is specified, then default to /NODE
: 2455 2571 2 |
: 2456 2572 2 | IF NOT (
: 2457 2573 2 | .FLAGS [FLAG_V_NODE] OR
: 2458 2574 2 | .FLAGS [FLAG_V_DEVICE] OR
: 2459 2575 2 | .FLAGS [FLAG_V_SERVERS]
: 2460 2576 2 | )
: 2461 2577 2 THEN
: 2462 2578 2 | FLAGS [FLAG_V_NODE] = TRUE
: 2463 2579 2 ;
: 2464 2580 2
: 2465 2581 1 END;
```

```
000C 00000 LCP_SHOWCOUQUAL:
53 00000000G 00 9E 00002 .WORD Save R2,R3 ; 2529
52 0000' CF 9E 00009 MOVAB CLISP$PRESENT, R3 ;
0000' CF 9F 0000E MOVAB FLAGS, R2 ;
63 01 FB 00012 PUSHAB QUAL_NODE ; 2565
02 50 FO 00015 CALLS #1, CLISP$PRESENT ;
0000' CF 9F 0001A INSV R0, #2, #1, FLAGS ;
63 01 FB 0001E PUSHAB QUAL_DEVICE ; 2566
01 50 FO 00021 CALLS #1, CLISP$PRESENT ;
0000' CF 9F 00026 INSV R0, #1, #1, FLAGS ;
63 01 FB 0002A PUSHAB QUAL_SERVERS ; 2567
00 50 FO 0002D CALLS #1, CLISP$PRESENT ;
62 02 E0 00032 INSV R0, #0, #1, FLAGS ;
62 01 E0 00036 BBS #2, FLAGS, 1$ ; 2573
03 62 E8 0003A BBS #1, FLAGS, 1$ ; 2574
62 04 88 0003D BLBS FLAGS, 1$ ; 2575
04 00040 1$: BISB2 #4, FLAGS ; 2578
RET ; 2581
```

; Routine Size: 65 bytes, Routine Base: \$CODE\$ + 0B74


```
2467 2582 1 %SBTTL 'LCP_SHOWCOU Process show counters request'
2468 2583 1 ROUTINE LCP_SHOWCOU : NOVALUE =
2469 2584 1
2470 2585 1 ++
2471 2586 1 FUNCTIONAL DESCRIPTION:
2472 2587 1
2473 2588 1 Convert and write the requested counters to sys$output.
2474 2589 1
2475 2590 1 FORMAL PARAMETERS:
2476 2591 1
2477 2592 1 NONE
2478 2593 1
2479 2594 1 IMPLICIT INPUTS:
2480 2595 1
2481 2596 1 counters in LAT driver or Ethernet device driver
2482 2597 1
2483 2598 1 IMPLICIT OUTPUTS:
2484 2599 1
2485 2600 1 NONE
2486 2601 1
2487 2602 1 ROUTINE VALUE:
2488 2603 1 COMPLETION CODES:
2489 2604 1
2490 2605 1 errors signalled
2491 2606 1
2492 2607 1 SIDE EFFECTS:
2493 2608 1
2494 2609 1 NONE
2495 2610 1
2496 2611 1 --
2497 2612 1
2498 2613 2 BEGIN
2499 2614 2
2500 2615 2
2501 2616 2 If the /NODE qualifier is present, then display Host Node counters.
2502 2617 2
2503 2618 2 IF .FLAGS [FLAG_V_NODE]
2504 2619 2 THEN
2505 2620 2 LCP_SHOWNODECOU ( )
2506 2621 2 ;
2507 2622 2
2508 2623 2
2509 2624 2 If the /DEVICE qualifier is present, then display Ethernet device
2510 2625 2 counters.
2511 2626 2
2512 2627 2 IF .FLAGS [FLAG_V_DEVICE]
2513 2628 2 THEN
2514 2629 2 LCP_SHOWDEVCOU ( )
2515 2630 2 ;
2516 2631 2
2517 2632 2
2518 2633 2 If the /SERVERS qualifier is present, then display SERVER counters.
2519 2634 2
2520 2635 2 IF .FLAGS [FLAG_V_SERVERS]
2521 2636 2 THEN
2522 2637 2 LCP_SHOWSERVCOU ( )
2523 2638 2 ;
```

0004 00000 LCP_SHOWCOU:
05 52 0000' CF 9E 00002 .WORD Save R2 : 2583
62 02 E1 00007 MOVAB FLAGS, R2 :
05 0000V CF 00 FB 0000B BBC #2, FLAGS, 1\$: 2618
62 01 E1 00010 1\$: CALLS #0, LCP_SHOWNODECOU : 2620
0000V CF 00 FB 00014 BBC #1, FLAGS, 2\$: 2627
05 62 E9 00019 2\$: CALLS #0, LCP_SHOWDEVCOU : 2629
0000V CF 00 FB 0001C BLBC FLAGS, 3\$: 2635
04 00021 3\$: CALLS #0, LCP_SHOWSERVCOU : 2637
RET : 2640

; Routine Size: 34 bytes,
Routine Base: \$CODE\$ + 0BB5


```
: 2527      2641 1 %SBTTL 'LCP_SHOWNODECOU Show Host Node counters'
: 2528      2642 1 ROUTINE LCP_SHOWNODECOU : NOVALUE =
: 2529      2643 1
: 2530      2644 1 ++
: 2531      2645 1 FUNCTIONAL DESCRIPTION:
: 2532      2646 1
: 2533      2647 1 Convert and write the NODE counters to sys$output.
: 2534      2648 1
: 2535      2649 1 FORMAL PARAMETERS:
: 2536      2650 1
: 2537      2651 1 NONE
: 2538      2652 1
: 2539      2653 1 IMPLICIT INPUTS:
: 2540      2654 1
: 2541      2655 1 counters in LAT driver
: 2542      2656 1
: 2543      2657 1 IMPLICIT OUTPUTS:
: 2544      2658 1
: 2545      2659 1 NONE
: 2546      2660 1
: 2547      2661 1 ROUTINE VALUE:
: 2548      2662 1 COMPLETION CODES:
: 2549      2663 1
: 2550      2664 1 NONE
: 2551      2665 1
: 2552      2666 1 SIDE EFFECTS:
: 2553      2667 1
: 2554      2668 1 NONE
: 2555      2669 1
: 2556      2670 1 --
: 2557      2671 1
: 2558      2672 2 BEGIN
: 2559      2673 2
: 2560      2674 2 LOCAL
: 2561      2675 2 LT : REF BLOCK [,BYTE],
: 2562      2676 2 PTR : REF VECTOR [],
: 2563      2677 2 STATUS
: 2564      2678 2 ;
: 2565      2679 2
: 2566      2680 2
: 2567      2681 2 CH$FILL (255, GHBS$COMM_AREA, AREA);
: 2568      2682 2
: 2569      2683 2
: 2570      2684 2 ! Get the counters if we can, and if not then signal the error
: 2571      2685 2 !
: 2572      2686 2 IF NOT
: 2573      2687 2 (STATUS = $CMKRN (ROUTIN = LCPK_GETCOUNTERS) )
: 2574      2688 2 THEN
: 2575      2689 2 BEGIN
: 2576      2690 2 SIGNAL (.STATUS);
: 2577      2691 2 RETURN
: 2578      2692 2 END
: 2579      2693 2 ;
: 2580      2694 2
: 2581      2695 2 !
: 2582      2696 2 ! convert the counters for output in a single bound
: 2583      2697 2
```

```
: 2584      2698 2  $SETDSC (OUTDSC, OUTBUFSIZE, OUTBUF);
: 2585      2699
: 2586      2700 STATUS = $FAOL
: 2587      2701 (
: 2588      2702   CTRSTR = COUNTERSCTRSTR,
: 2589      2703   OUTLEN = OUTDSC [DSC$W_LENGTH],
: 2590      2704   OUTBUF = OUTDSC,
: 2591      2705   PRMLST = AREA [GHB$T_COUNTERS]
: 2592      2706 );
: 2593      2707
: 2594      2708 IF NOT .STATUS THEN SIGNAL (.STATUS);
: 2595      2709
: 2596      2710 STATUS = LIB$PUT_OUTPUT (OUTDSC);
: 2597      2711
: 2598      2712 IF NOT .STATUS THEN SIGNAL (.STATUS);
: 2599      2713
: 2600      2714
: 2601      2715 : Print the mask bit names that are present
: 2602      2716
: 2603      2717 LT = AREA;
: 2604      2718 IF .LT [GHB$L_PROTOMASK] NEQ 0
: 2605      2719 THEN
: 2606      2720 BEGIN
: 2607      2721   STATUS = LIB$PUT_OUTPUT (PROTOCOLMASK);
: 2608      2722   PTR = PROTOCOLBITS;
: 2609      2723   INCR IDX FROM 0 TO 64 BY 2
: 2610      2724 DO
: 2611      2725   BEGIN
: 2612      2726   : If the string desc is blank, then exit loop
: 2613      2727   :
: 2614      2728   IF .PTR [.IDX + 1] EQL 0
: 2615      2729   THEN
: 2616      2730   EXITLOOP
: 2617      2731   :
: 2618      2732   : If the bit is set, then write the string
: 2619      2733   :
: 2620      2734   IF .LT [$BYTEOFFSET (GHB$L_PROTOMASK), .PTR [.IDX], 1, 0]
: 2621      2735   THEN
: 2622      2736   LIB$PUT_OUTPUT (.PTR [.IDX + 1] )
: 2623      2737   :
: 2624      2738   END
: 2625      2739
: 2626      2740 : END
: 2627      2741
: 2628      2742
: 2629      2743
: 2630      2744 1  END;
```

01FC 00000 LCP_SHOWNODECOU:

58	00000000G	00	9E	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8	: 2642
57	00000000G	00	9E	00009	MOVAB	LIB\$PUT_OUTPUT, R8	:
56	0000	CF	9E	00010	MOVAB	LIB\$SIGNAL, R7	:
					MOVAB	OUTDSC, R6	:

LATCP
V04-000

LAT Control Program
LCP_SHOWNODECOU Show Host Node counters

H 11
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 86
(24)

0060	8F	FF	8F	6E		00	2C	00015	MOV C5	#0, (SP), #255, #96, AREA	:	2681
				F5FC	C6	02	D4	0001D	CLRL	-(SP)	:	2687
				0000V	7E	02	9F	00020	PUSHAB	LCPK_GETCOUNTERS	:	
		00000000G	00		CF	02	FB	00022	CALLS	#2, SYSSCMKRN	:	
			52		50	50	D0	0002D	MOVL	R0, STATUS	:	
			06		52	52	E8	00030	BLBS	STATUS, 1\$	:	
					52	DD	00033	PUSHL	STATUS		:	2690
			67		01	FB	00035	CALLS	#1, LIB\$SIGNAL		:	2689
					04	04	00038	RET			:	2698
			66	0400	8F	3C	00039	1\$:	MOVZWL	#1024, OUTDSC	:	
		04	A6	08	A6	9E	0003E		MOVAB	OUTBUF, OUTDSC+4	:	
				F628	C6	9F	00043		PUSHAB	AREA+44	:	2706
					56	DD	00047		PUSHL	R6	:	
					56	DD	00049		PUSHL	R6	:	
		00000000G	00	0000'	CF	9F	0004B		PUSHAB	COUNTERSCTRSTR	:	
			52		04	FB	0004F		CALLS	#4, SYSSFAOL	:	
			05		50	D0	00056		MOVL	R0, STATUS	:	
					52	E8	00059		BLBS	STATUS, 2\$	:	2708
			67		52	DD	0005C		PUSHL	STATUS	:	
					01	FB	0005E		CALLS	#1, LIB\$SIGNAL	:	
			68		56	DD	00061	2\$:	PUSHL	R6	:	2710
			52		01	FB	00063		CALLS	#1, LIB\$PUT_OUTPUT	:	
			05		50	D0	00066		MOVL	R0, STATUS	:	
					52	E8	00069		BLBS	STATUS, 3\$	:	2712
			67		52	DD	0006C		PUSHL	STATUS	:	
			53	F5FC	01	FB	0006E		CALLS	#1, LIB\$SIGNAL	:	
				50	C6	9E	00071	3\$:	MOVAB	AREA, LT	:	2717
					A3	D5	00076		TSTL	80(LT)	:	2718
					2D	13	00079		BEQL	6\$	:	
			68	0000'	CF	9F	0007B		PUSHAB	PROTOCOLMASK	:	2721
			52		01	FB	0007F		CALLS	#1, LIB\$PUT_OUTPUT	:	
			54	0000'	50	D0	00082		MOVL	R0, STATUS	:	
					CF	9E	00085		MOVAB	PROTOCOLBITS, PTR	:	2722
					52	D4	0008A		CLRL	IDX	:	2729
			50	04	A442	D0	0008C	4\$:	MOVL	4(PTR)[IDX], R0	:	
					15	13	00091		BEQL	6\$	:	
		05	50	A3	6442	E1	00093		BBC	(PTR)[IDX], 80(LT), 5\$	:	2736
					50	DD	00099		PUSHL	R0	:	2738
			68		01	FB	0009B		CALLS	#1, LIB\$PUT_OUTPUT	:	
	FFE4		52		02	00000040	8F	F1	0009E	5\$:	:	2736
					04	000AB	6\$:	RET			:	2744

; Routine Size: 169 bytes, Routine Base: \$CODE\$ + 0BD7

; 2631 2745 1

LATCP
V04-000

LAT Control Program
LCP_SHOWDEVCOU Show Ethernet device counters

I 11
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1 Page 87
(25)

```
2633 2746 1 %SBTTL 'LCP_SHOWDEVCOU Show Ethernet device counters'
2634 2747 1 ROUTINE LCP_SHOWDEVCOU : NOVALUE =
2635 2748 1
2636 2749 1 ++
2637 2750 1 FUNCTIONAL DESCRIPTION:
2638 2751 1
2639 2752 1 Convert and write the Ethernet device counters to sys$output.
2640 2753 1
2641 2754 1 FORMAL PARAMETERS:
2642 2755 1
2643 2756 1 NONE
2644 2757 1
2645 2758 1 IMPLICIT INPUTS:
2646 2759 1
2647 2760 1 counters in LAT driver
2648 2761 1
2649 2762 1 IMPLICIT OUTPUTS:
2650 2763 1
2651 2764 1 NONE
2652 2765 1
2653 2766 1 ROUTINE VALUE:
2654 2767 1 COMPLETION CODES:
2655 2768 1
2656 2769 1 NONE
2657 2770 1
2658 2771 1 SIDE EFFECTS:
2659 2772 1
2660 2773 1 NONE
2661 2774 1
2662 2775 1 --
2663 2776 1
2664 2777 2 BEGIN
2665 2778 2
2666 2779 2 LITERAL
2667 2780 2 CTRBUFSIZE = 512
2668 2781 2 ;
2669 2782 2
2670 2783 2 LOCAL
2671 2784 2 CTRDSC : BLOCK [DSC$K_Z_BLN, BYTE],
2672 2785 2 STATUS
2673 2786 2 ;
2674 2787 2
2675 2788 2
2676 2789 2 ! Setup the counters return buffer descriptor
2677 2790 2 !
2678 2791 2 $SETDSC (CTRDSC, CTRBUFSIZE, AREA);
2679 2792 2 !
2680 2793 2 !
2681 2794 2 Read and display the counters
2682 2795 2 !
2683 2796 3 IF NOT (STATUS = $CMKRNL (ROUTIN = LCPK_SHOWDEVCOU) )
2684 2797 2 OR
2685 2798 3 NOT (STATUS = LCP_DISPDEVCOU (CTRDSC) )
2686 2799 2 THEN
2687 2800 2 SIGNAL (.STATUS)
2688 2801 2 ;
2689 2802 2
```


LATCP
V04-000
; 2690

LAT Control Program
LCP_SHOWDEVCOU Show Ethernet device counters
2803 1 END;

J 11
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1
Page 88
(25)

			0000 00000 LCP_SHOWDEVCOU:			
	5E		04 C2 00002	.WORD	Save nothing	: 2747
	7E	0200	8F 3C 00005	SUBL2	#4, SP	: 2791
04	AE	0000'	CF 9E 0000A	MOVZWL	#512, CTRDSC	: 2796
			7E D4 00010	MOVAB	AREA, CTRDSC+4	
		0000V	CF 9F 00012	CLRL	-(SP)	
00000000G	00		02 FB 00016	PUSHAB	LCPK_SHOWDEVCOU	
	0A		50 E9 0001D	CALLS	#2, SYSSCMKRN	
			5E DD 00020	BLBC	STATUS, 1\$	: 2798
0000V	CF		01 FB 00022	PUSHL	SP	
	09		50 E8 00027	CALLS	#1, LCP_DISPDEVCOU	
			50 DD 0002A 1\$:	BLBS	STATUS, -2\$	: 2800
00000000G	00		01 FB 0002C	PUSHL	STATUS	: 2803
			04 00033 2\$:	CALLS	#1, LIB\$SIGNAL	
				RET		

; Routine Size: 52 bytes, Routine Base: \$CODE\$ + 0C80

```
2692 2804 1 %SBTTL 'LCPK_SHOWDEVCOU Show Ethernet device counters'
2693 2805 1 ROUTINE LCPK_SHOWDEVCOU (CTRDSC) =
2694 2806 1
2695 2807 1 ++
2696 2808 1 FUNCTIONAL DESCRIPTION:
2697 2809 1
2698 2810 1 Convert and write the Ethernet device counters to sys$output.
2699 2811 1
2700 2812 1 FORMAL PARAMETERS:
2701 2813 1
2702 2814 1 CTRDSC Address of counter block descriptor
2703 2815 1
2704 2816 1 IMPLICIT INPUTS:
2705 2817 1
2706 2818 1 counters in Ethernet driver
2707 2819 1
2708 2820 1 IMPLICIT OUTPUTS:
2709 2821 1
2710 2822 1 NONE
2711 2823 1
2712 2824 1 ROUTINE VALUE:
2713 2825 1 COMPLETION CODES:
2714 2826 1
2715 2827 1 error values returned
2716 2828 1
2717 2829 1 SIDE EFFECTS:
2718 2830 1
2719 2831 1 NONE
2720 2832 1
2721 2833 1 --
2722 2834 1
2723 2835 2 BEGIN
2724 2836 2
2725 2837 2 LITERAL
2726 2838 2 DEVNAMSIZE = 16,
2727 2839 2 CTRBUFSIZE = 512
2728 2840 2 ;
2729 2841 2
2730 2842 2 BIND
2731 2843 2 DEVNAMCTRL = DESCRIPTOR ('!AC!UW')
2732 2844 2 ;
2733 2845 2
2734 2846 2 LOCAL
2735 2847 2 LT : REF BLOCK [,BYTE],
2736 2848 2 DEVNAMSTR : VECTOR [DEVNAMSIZE, BYTE],
2737 2849 2 DEVNAMDSC : BLOCK [DSC$K_Z_BLN, BYTE],
2738 2850 2 RDCTRDSC : BLOCK [DSC$K_Z_BLN, BYTE],
2739 2851 2 UCB : REF BLOCK [,BYTE],
2740 2852 2 ORB : REF BLOCK [,BYTE],
2741 2853 2 DDB : REF BLOCK [,BYTE],
2742 2854 2 IOSB : VECTOR [4, WORD],
2743 2855 2 STATUS,
2744 2856 2 DEVCHAN : WORD,
2745 2857 2 PID,
2746 2858 2 OWNUIC,
2747 2859 2 VPROT : WORD
2748 2860 2 ;
```



```
2749 2861 2
2750 2862 2
2751 2863 2      | If the data link UCB is gone, then we are inactive
2752 2864 2
2753 2865 2      | LT = .LCPSL_AREA;
2754 2866 2      | IF (UCB = .[T [GHB$$_UCB] ) EQL 0
2755 2867 2      | THEN
2756 2868 2      |     RETURN SSS_DEVINACT
2757 2869 2
2758 2870 2
2759 2871 2      | Get pointers to the ucb and ddb for datalink to make its name
2760 2872 2      | so we can assign a channel to the data link device ucb
2761 2873 2
2762 2874 2      | DDB = .UCB [UCB$$_DDB];
2763 2875 2      | DEVNAMDSC [DSC$$_POINTER] = DEVNAMSTR;
2764 2876 2      | DEVNAMDSC [DSC$$_LENGTH] = DEVNAMSIZE;
2765 2877 2      | STATUS = $FAO
2766 2878 2      | (
2767 2879 2      |     DEVNAMCTRL
2768 2880 2      |     DEVNAMDSC [DSC$$_LENGTH],
2769 2881 2      |     DEVNAMDSC
2770 2882 2      |     DDB [DDB$$_NAME],
2771 2883 2      |     .UCB [UCB$$_UNIT]
2772 2884 2      | );
2773 2885 2      | IF NOT .STATUS
2774 2886 2      | THEN
2775 2887 2      |     RETURN .STATUS
2776 2888 2
2777 2889 2
2778 2890 2      | Fix the ucb so we can assign a channel to it... save current
2779 2891 2      | settings first.
2780 2892 2
2781 2893 2      | PID = .UCB [UCB$$_PID];
2782 2894 2      | ORB = .UCB [UCB$$_ORB];
2783 2895 2      | OWNUIC = .ORB [ORB$$_OWNER];
2784 2896 2      | VPROT = .ORB [ORB$$_PROT];
2785 2897 2      | UCB [UCB$$_PID] = 0;
2786 2898 2      | ORB [ORB$$_OWNER] = 0;
2787 2899 2      | ORB [ORB$$_PROT] = 0;
2788 2900 2
2789 2901 2      | Assign a channel to hold it here and allow us to shut it down
2790 2902 2      | later and deassign the ucb to cause it to go away.
2791 2903 2
2792 2904 2      | STATUS = $ASSIGN (DEVNAM = DEVNAMDSC, CHAN = DEVCHAN);
2793 2905 2      | IF NOT .STATUS
2794 2906 2      | THEN
2795 2907 2      |     RETURN .STATUS
2796 2908 2
2797 2909 2
2798 2910 2
2799 2911 2      | Setup the counters return buffer descriptor
2800 2912 2
2801 2913 2      | $SETDSC (RDCTRDSC, CTRBUFSIZE, AREA);
2802 2914 2
2803 2915 2
2804 2916 2      | Read the counters from the Ethernet device
2805 2917 2
```

```

: 2806 P 2918 2 STATUS = $QIOW
: 2807 P 2919 (
: 2808 P 2920 FUNC = IOS$SENSEMODE OR IOSM_CTRL OR IOSM_RD_COUNT,
: 2809 P 2921 CHAN = .DEVCHAN,
: 2810 P 2922 IOSB = IOSB,
: 2811 P 2923 P2 = RDCTRDSC
: 2812 2924 );
: 2813 2925
: 2814 2926 !
: 2815 2927 ! Restore device settings
: 2816 2928
: 2817 2929 UCB [UCB$L_PID] = .PID;
: 2818 2930 ORB [ORB$L_OWNER] = .OWNUIC;
: 2819 2931 ORB [ORB$W_PROT] = .VPROT;
: 2820 2932
: 2821 2933 !
: 2822 2934 ! Deassign the channel that we have.
: 2823 2935
: 2824 2936 $DASSGN (CHAN = .DEVCHAN);
: 2825 2937
: 2826 2938 RETURN .STATUS ! Return the status from the READ QIO
: 2827 2939
: 2828 2940 1
END;
```

```

.PSECT $PLITS$,NOWRT,NOEXE,2
57 55 21 43 41 21 00E00 P.AFP: .ASCII \!AC!UW\
00E06 .BLKB 2
00000006 00E08 P.AFO: .LONG 6
00000000 00E0C .ADDRESS P.AFP
DEVNAMCTRL= P.AFO
```

```

.PSECT $CODE$,NOWRT,2
00FC 0000 LCPK_SHOWDEVCOU:
5E 0000' 2C C2 00002 .WORD Save R2,R3,R4,R5,R6,R7
50 14 CF D0 00005 SUBL2 #44, SP
53 20D4 A0 D0 0000A MOVL LCP$L_AREA, LT
50 8F 06 12 0000E MOVL 20(LT), UCB
50 04 00015 BNEQ 1$
18 50 28 A3 D0 00016 1$: MOVL 40(UCB), DDB
14 AE 1C AE 9E 0001A MOVAB DEVNAMSTR, DEVNAMDSC+4
14 AE 10 B0 0001F MOVW #16, DEVNAMDSC
7E 54 A3 3C 00023 MOVZWL 84(UCB), -(SP)
14 A0 9F 00027 PUSHAB 20(DDb)
1C AE 9F 0002A PUSHAB DEVNAMDSC
20 AE 9F 0002D PUSHAB DEVNAMDSC
0000' CF 9F 00030 PUSHAB DEVNAMCTRL
00000000G 00 05 FB 00034 CALLS #5, SYSS$FAO
54 50 D0 0003B MOVL R0, STATUS
70 54 E9 0003E BLBC STATUS, 2$
```


LATCP
V04-000

LAT Control Program
LCPK_SHOWDEVCOU Show Ethernet device counters

N 11
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 92
(26)

	57	2C	A3	D0	00041	MOVL	44(UCB), PID	:	2893
	52	1C	A3	D0	00045	MOVL	28(UCB), ORB	:	2894
	56		62	D0	00049	MOVL	(ORB), OWNUIC	:	2895
	55	18	A2	B0	0004C	MOVW	24(ORB), VPROT	:	2896
		2C	A3	D4	00050	CLRL	44(UCB)	:	2897
			62	D4	00053	CLRL	(ORB)	:	2898
		18	A2	B4	00055	CLRW	24(ORB)	:	2899
			7E	7C	00058	CLRQ	-(SP)	:	2904
		08	AE	9F	0005A	PUSHAB	DEVCHAN	:	
		20	AE	9F	0005D	PUSHAB	DEVNAMDSC	:	
00000000G	00		04	FB	00060	CALLS	#4, SYSS\$ASSIGN	:	
	54		50	D0	00067	MOVL	R0, STATUS	:	
	44		54	E9	0006A	BLBC	STATUS, 2\$	:	2905
0C	AE	0200	8F	3C	0006D	MOVZWL	#512, RDCTRDSC	:	2913
10	AE	0000	CF	9E	00073	MOVAB	AREA, RDCTRDSC+4	:	
			7E	7C	00079	CLRQ	-(SP)	:	2924
			7E	7C	0007B	CLRQ	-(SP)	:	
		1C	AE	9F	0007D	PUSHAB	RDCTRDSC	:	
			7E	7C	00080	CLRQ	-(SP)	:	
			7E	D4	00082	CLRL	-(SP)	:	
		24	AE	9F	00084	PUSHAB	IOSB	:	
	7E	0327	8F	3C	00087	MOVZWL	#807, -(SP)	:	
	7E	28	AE	3C	0008C	MOVZWL	DEVCHAN, -(SP)	:	
			7E	D4	00090	CLRL	-(SP)	:	
00000000G	00		0C	FB	00092	CALLS	#12, SYSS\$QIOW	:	
	54		50	D0	00099	MOVL	R0, STATUS	:	
2C	A3		57	D0	0009C	MOVL	PID, 44(UCB)	:	2929
	62		56	D0	000A0	MOVL	OWNUIC, (ORB)	:	2930
18	A2		55	B0	000A3	MOVW	VPROT, 24(ORB)	:	2931
	7E		6E	3C	000A7	MOVZWL	DEVCHAN, -(SP)	:	2936
00000000G	00		01	FB	000AA	CALLS	#1, SYSS\$DASSGN	:	
	50		54	D0	000B1	MOVL	STATUS, R0	:	2938
			04	000B4	2\$: RET			:	2940

; Routine Size: 181 bytes, Routine Base: \$CODE\$ + 0CB4

LATCP
V04-000

LAT Control Program

LCP_DISPDEVCOU Display the Ethernet device cou

B 12

16-Sep-1984 01:49:16

14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742

DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 93

(27)

```
2830 2941 1 %SBTTL 'LCP_DISPDEVCOU Display the Ethernet device counters'
2831 2942 1 ROUTINE LCP_DISPDEVCOU (CTRDSC) =
2832 2943 1
2833 2944 1 ++
2834 2945 1 FUNCTIONAL DESCRIPTION:
2835 2946 1
2836 2947 1 Convert and write the all the Ethernet device counters to sys$output.
2837 2948 1
2838 2949 1 FORMAL PARAMETERS:
2839 2950 1
2840 2951 1 CTRDSC Address of descriptor of the counters string
2841 2952 1
2842 2953 1 IMPLICIT INPUTS:
2843 2954 1
2844 2955 1 NONE
2845 2956 1
2846 2957 1 IMPLICIT OUTPUTS:
2847 2958 1
2848 2959 1 NONE
2849 2960 1
2850 2961 1 ROUTINE VALUE:
2851 2962 1 COMPLETION CODES:
2852 2963 1
2853 2964 1 error values returned
2854 2965 1
2855 2966 1 SIDE EFFECTS:
2856 2967 1
2857 2968 1 NONE
2858 2969 1
2859 2970 1 --
2860 2971 1
2861 2972 2 BEGIN
2862 2973 2
2863 2974 2 LITERAL PARMSIZE = 50;
2864 2975 2
2865 2976 2 LOCAL
2866 2977 2 PARAM : VECTOR [PARMSIZE, LONG],
2867 2978 2 PINDX,
2868 2979 2 CTRID,
2869 2980 2 STATUS,
2870 2981 2 VALUE,
2871 2982 2 BITMAP,
2872 2983 2 RCV_BITMAP : BLOCK [4, BYTE],
2873 2984 2 XMT_BITMAP : BLOCK [4, BYTE],
2874 2985 2 PTR : REF VECTOR []
2875 2986 2 ;
2876 2987 2
2877 2988 2 RCV_BITMAP = 0;
2878 2989 2 XMT_BITMAP = 0;
2879 2990 2
2880 2991 2 Loop through all the counters
2881 2992 2
2882 2993 2 PINDX = 0;
2883 2994 2 CTRID = CTRNAMES;
2884 2995 2 WHILE ..CTRID NEQ -1
2885 2996 2 DO
2886 2997 3 BEGIN
```



```
2887 2998      |
2888 2999      |      Get the counter values
2889 3000      |
2890 3001      | LCP_FINDCOU (.CTRDSC, ..CTRID, VALUE, BITMAP);
2891 3002      | PARAM [.PINDEX] = .VALUE;
2892 3003      | SELECTONEU ..CTRID OF
2893 3004      | SET
2894 3005      |
2895 3006      | [NMASC_CTLIN_RFL] :      ! If receive failures
2896 3007      | (
2897 3008      |     RCV_BITMAP = .BITMAP      ! Save the bitmap
2898 3009      | );
2899 3010      |
2900 3011      | [NMASC_CTLIN_SFL] :      ! Is send failures
2901 3012      | (
2902 3013      |     XMT_BITMAP = .BITMAP      ! Save the bitmap
2903 3014      | );
2904 3015      |
2905 3016      | TES
2906 3017      | ;
2907 3018      |
2908 3019      | PINDEX = .PINDEX + 1;      ! Skip to next parameter
2909 3020      | CTRID = .CTRID + 4      ! Skip to next counter ID
2910 3021      |
2911 3022      | END
2912 3023      |
2913 3024      | ;
2914 3025      |
2915 3026      |      Convert the counters for output
2916 3027      |
2917 3028      | $SETDSC (OUTDSC, OUTBUFSIZE, OUTBUF);
2918 3029      |
2919 3030      | STATUS = $FAOL
2920 3031      | (
2921 3032      |     CTRSTR = DEVCTRSTR,
2922 3033      |     OUTLEN = OUTDSC [DSC$W_LENGTH],
2923 3034      |     OUTBUF = OUTDSC,
2924 3035      |     PRMLST = PARAM[0]
2925 3036      | );
2926 3037      |
2927 3038      | IF NOT .STATUS THEN RETURN .STATUS;
2928 3039      |
2929 3040      | STATUS = LIB$PUT_OUTPUT (OUTDSC);
2930 3041      |
2931 3042      | IF NOT .STATUS THEN RETURN .STATUS;
2932 3043      |
2933 3044      |      Print the receive mask bit names that are present
2934 3045      |
2935 3046      |      IF .RCV_BITMAP [0, 0, 16, 0] NEQ 0
2936 3047      |      THEN
2937 3048      |      BEGIN
2938 3049      |      |
2939 3050      |      |      Print the header
2940 3051      |      |
2941 3052      |      |
2942 3053      |      | STATUS = LIB$PUT_OUTPUT (RCVERRMASK);
2943 3054      |      | PTR = RCVERRBITS;      ! Point to the translation tbl
```

```

2944 3055 3      INCR INDX FROM 0 TO 32 BY 2
2945 3056 3      DO
2946 3057 4          BEGIN
2947 3058 4              If the string desc is blank, then exit loop
2948 3059 4              If .PTR [.INDX + 1] EQL 0
2949 3060 4              THEN
2950 3061 4                  EXITLOOP
2951 3062 4              :
2952 3063 4              If the bit is set, then write the string
2953 3064 4              If .RCV_BITMAP [0, .PTR [.INDX], 1, 0]
2954 3065 4              THEN
2955 3066 4                  LIB$PUT_OUTPUT (.PTR [.INDX + 1] )
2956 3067 4              END
2957 3068 4          :
2958 3069 4          END
2959 3070 4      :
2960 3071 4      END
2961 3072 3      :
2962 3073 3      :
2963 3074 2      :
2964 3075 2      :
2965 3076 2      :
2966 3077 2      Print the transmit mask bit names that are present
2967 3078 2      :
2968 3079 2      IF .XMT_BITMAP [0, 0, 16, 0] NEQ 0
2969 3080 2      THEN
2970 3081 3          BEGIN
2971 3082 3              Print the header
2972 3083 3              :
2973 3084 3              STATUS = LIB$PUT_OUTPUT (XMTERRMASK);
2974 3085 3              PTR = XMTERRBITS; ! Point to the translation tbl
2975 3086 3              INCR INDX FROM 0 TO 32 BY 2
2976 3087 3              DO
2977 3088 4                  BEGIN
2978 3089 4                      If the string desc is blank, then exit loop
2979 3090 4                      If .PTR [.INDX + 1] EQL 0
2980 3091 4                      THEN
2981 3092 4                          EXITLOOP
2982 3093 4                      :
2983 3094 4                      If the bit is set, then write the string
2984 3095 4                      If .XMT_BITMAP [0, .PTR [.INDX], 1, 0]
2985 3096 4                      THEN
2986 3097 4                          LIB$PUT_OUTPUT (.PTR [.INDX + 1] )
2987 3098 4                      END
2988 3099 4                  :
2989 3100 4                  END
2990 3101 4              :
2991 3102 4              END
2992 3103 3          :
2993 3104 3          END
2994 3105 3      :
2995 3106 3      :
2996 3107 2      :
2997 3108 2      SUCCESS
2998 3109 2      :
2999 3110 1      END;
```



```
01FC 00000 LCP_DISPDEVCOU:
      58      0000' CF 9E 00002      .WORD      Save R2,R3,R4,R5,R6,R7,R8      : 2942
      57 00000000G 00 9E 00007      MOVAB      OUTDSC, R8
      5E      FF30 CE 9E 0000E      MOVAB      LIB$PUT_OUTPUT, R7
      55      7C 00013      MOVAB      -208(SP), SP
      52      D4 00015      CLRQ      XMT_BITMAP      : 2989
      53      0000' CF 9E 00017      CLRL      PINDX      : 2993
      8F      63 D1 0001C 1$:      MOVAB      CTRNAMES, CTRID      : 2994
      36      13 00023      CMPL      (CTRID), #-1      : 2995
      5E      DD 00025      BEQL      4$
      08      AE 9F 00027      PUSHL     SP      : 3001
      63      DD 0002A      PUSHAB    VALUE
      04      AC DD 0002C      PUSHL     (CTRID)
      04      AE FB 0002F      PUSHL     CTRDSC
      0000V CF 04      04      FB 0002F      CALLS     #4, LCP_FINDCOU
      08 AE42      04      AE D0 00034      MOVL      VALUE, PARAM[PINDX]      : 3002
      00000426 8F      63 D1 0003A      CMPL      (CTRID), #1062      : 3006
      56      05 12 00041      BNEQ      2$
      00000424 8F      6E D0 00043      MOVL      BITMAP, RCV_BITMAP      : 3008
      0C      11 00046      BRB       3$
      63 D1 00048 2$:      CMPL      (CTRID), #1060      : 3007
      03      12 0004F      BNEQ      3$
      55      6E D0 00051      MOVL      BITMAP, XMT_BITMAP      : 3013
      52      D6 00054 3$:      INCL     PINDX      : 3019
      53      04 C0 00056      ADDL2    #4, CTRID      : 3020
      68      C1 11 00059      BRB       1$
      04      8F 3C 0005B 4$:      MOVZWL   #1024, OUTDSC      : 3028
      08      A8 9E 00060      MOVAB     OUTBUF, OUTDSC+4
      08      AE 9F 00065      MOVAB     PARAM
      58      DD 00068      PUSHL     R8      : 3036
      58      DD 0006A      PUSHL     R8
      00000000G 00 0000' CF 9F 0006C      PUSHAB    DEVCTRSTR
      54      04 FB 00070      CALLS     #4, SYSS$FAOL
      0B      50 D0 00077      MOVL      R0, STATUS
      58      54 E9 0007A      BLBC     STATUS, 5$      : 3038
      67      01 FB 0007F      PUSHL     R8      : 3040
      54      50 D0 00082      CALLS     #1, LIB$PUT_OUTPUT
      04      54 E8 00085      MOVL      R0, STATUS
      50      54 D0 00088 5$:      BLBS     STATUS, 6$      : 3042
      04      04 0008B      MOVL      STATUS, R0
      56      B5 0008C 6$:      RET
      28      13 0008E      TSTW     RCV_BITMAP      : 3047
      0000' CF 9F 00090      BEQL     9$
      67      01 FB 00094      PUSHAB    RCVERRMASK      : 3053
      54      50 D0 00097      CALLS     #1, LIB$PUT_OUTPUT
      53      0000' CF 9E 0009A      MOVL      R0, STATUS
      52      D4 0009F      MOVAB     RCVERRBITS, PTR      : 3054
      50      04 A342 D0 000A1 7$:      CLRL     INDX      : 3061
      10      13 000A6      MOVL      4(PTR)[INDX], R0
      05      56      6342 E1 000A8      BEQL     9$
      50      DD 000AD      BBC      (PTR)[INDX], RCV_BITMAP, 8$      : 3068
      67      01 FB 000AF      PUSHL     R0      : 3070
      CALLS     #1, LIB$PUT_OUTPUT
```

LATCP
V04-000

LAT Control Program
LCP_DISPDEVCOU Display the Ethernet device cou

F 12
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 97
(27)

FFE9	52	02	20	F1	000B2	8\$:	ACBL	#32, #2, INDX, 7\$	3068
			55	B5	000B8	9\$:	TSTW	XMT_BITMAP	3079
			28	13	000BA		BEQL	12\$	
		0000'	CF	9F	000BC		PUSHAB	XMTERRMASK	3085
	67		01	FB	000C0		CALLS	#1, LIB\$PUT_OUTPUT	
	54		50	D0	000C3		MOVL	R0, STATUS	
	53	0000'	CF	9E	000C6		MOVAB	XMTERRBITS, PTR	3086
			54	D4	000CB		CLRL	INDX	3093
	50	04	A344	D0	000CD	10\$:	MOVL	4(PTR)[INDX], R0	
			10	13	000D2		BEQL	12\$	
	05		6344	E1	000D4		BBC	(PTR)[INDX], XMT_BITMAP, 11\$	3100
			50	DD	000D9		PUSHL	R0	3102
		67	01	FB	000DB		CALLS	#1, LIB\$PUT_OUTPUT	
FFE9	54	02	20	F1	000DE	11\$:	ACBL	#32, #2, INDX, 10\$	3100
		50	01	D0	000E4	12\$:	MOVL	#1, R0	3110
			04	000E7			RET		

; Routine Size: 232 bytes, Routine Base: \$CODE\$ + 0D69


```
3001 3111 1 %SBTTL 'LCP_FINDCOU Find the specific device counter'
3002 3112 1 ROUTINE LCP_FINDCOU (CTRDSC, CTRNAME, VALUE, BITMAP) =
3003 3113 1
3004 3114 1 ++
3005 3115 1 FUNCTIONAL DESCRIPTION:
3006 3116 1
3007 3117 1 Convert and write the all the Ethernet device counters to sys$output.
3008 3118 1
3009 3119 1 FORMAL PARAMETERS:
3010 3120 1
3011 3121 1 CTRDSC Address of descriptor of the counters string
3012 3122 1 CTRNAME Name (ID) of the counter id code to search for
3013 3123 1 VALUE Returned longword value for the counter if found
3014 3124 1 BITMAP Returned longword value for the bitmap if found
3015 3125 1
3016 3126 1 IMPLICIT INPUTS:
3017 3127 1
3018 3128 1 NONE
3019 3129 1
3020 3130 1 IMPLICIT OUTPUTS:
3021 3131 1
3022 3132 1 NONE
3023 3133 1
3024 3134 1 ROUTINE VALUE:
3025 3135 1 COMPLETION CODES:
3026 3136 1
3027 3137 1 error values returned
3028 3138 1
3029 3139 1 SIDE EFFECTS:
3030 3140 1
3031 3141 1 NONE
3032 3142 1
3033 3143 1 --
3034 3144 1
3035 3145 2 BEGIN
3036 3146 2
3037 3147 2 MAP
3038 3148 2 CTRDSC : REF BLOCK [,BYTE]
3039 3149 2 ;
3040 3150 2
3041 3151 2 LOCAL
3042 3152 2 CTRLLEN,
3043 3153 2 CTRPTR : REF BLOCK [,BYTE],
3044 3154 2 STATUS,
3045 3155 2 COUCOD,
3046 3156 2 COUVAL,
3047 3157 2 COUID : BLOCK [4, BYTE],
3048 3158 2 ERRMAP
3049 3159 2 ;
3050 3160 2
3051 3161 2 CTRPTR = .CTRDSC [DSC$A_POINTER];
3052 3162 2 CTRLLEN = .CTRDSC [DSC$W_LENGTH];
3053 3163 2 COUCOD = -1; ! Set to illegal value
3054 3164 2 WHILE .CTRLLEN GTR 0 AND
3055 3165 2 .COUCOD NEQ .CTRNAME
3056 3166 2 DO
3057 3167 2 BEGIN
```

```

3058      COUID = .CTRPTR [0, 0, 16, 0];      ! Get the counter id + bits
3059      COUCOD = .COUID [NMA$V_CNT_TYP];      ! Get just the counter code
3060      CTRPTR = .CTRPTR + 2;                  ! Skip counter id field
3061      CTRLN = .CTRLN - 2;                    ! Consume byte count
3062
3063      !
3064      ! Get the bitmap if present
3065      !
3066      IF .COUID [NMA$V_CNT_MAP]
3067      THEN
3068      BEGIN
3069          ERRMAP = .CTRPTR [0, 0, 16, 0];      ! Get the bitmap
3070          CTRPTR = .CTRPTR + 2;                  ! Skip it
3071          CTRLN = .CTRLN - 2;                    ! Consume bytes
3072      END
3073      ELSE
3074          ERRMAP = 0
3075      ;
3076
3077      !
3078      ! Dispatch on counter width
3079      !
3080      SELECTONEU .COUID [NMA$V_CNT_WID] OF
3081      SET
3082      [0] :                                     ! Unknown???
3083      (
3084          RETURN (LCP$_INTERNAL)
3085      );
3086      [1] :                                     ! 8 bit counter
3087      (
3088          COUVAL = .CTRPTR [0, 0, 8, 0];
3089          CTRPTR = .CTRPTR + 1
3090      );
3091      [2] :                                     ! 16 bit counter
3092      (
3093          COUVAL = .CTRPTR [0, 0, 16, 0];
3094          CTRPTR = .CTRPTR + 2
3095      );
3096      [3] :                                     ! 32 bit counter
3097      (
3098          COUVAL = .CTRPTR [0, 0, 32, 0];
3099          CTRPTR = .CTRPTR + 4
3100      );
3101      TES
3102      ;
3103      END
3104      ;
3105      IF .COUCOD NEQ .CTRNAME
3106      THEN
3107      BEGIN

```



```

: 3115      3225      3      .VALUE = 0;
: 3116      3226      3      .BITMAP = 0;
: 3117      3227      3      STATUS = FAILURE
: 3118      3228      3      END
: 3119      3229      3      ELSE
: 3120      3230      3      BEGIN
: 3121      3231      3      .VALUE = .COUVAL;
: 3122      3232      3      .BITMAP = .ERRMAP;
: 3123      3233      3      STATUS = SUCCESS
: 3124      3234      3      END
: 3125      3235      3      ;
: 3126      3236      3      RETURN .STATUS
: 3127      3237      3
: 3128      3238      3
: 3129      3239      3      END;

```

```

                                003C 00000 LCP_FINDCOU:
                                .WORD      Save R2,R3,R4,R5
                                50          04 AC D0 00002      MOVL      CTRDSC, R0
                                51          04 AO D0 00006      MOVL      4(R0), CTRPTR
                                54          60 3C 0000A      MOVZWL   (R0), CTRLEN
                                53          01 CE 0000D      MNEGL     #1, COUCOD
                                54          D5 00010 1$:      TSTL      CTRLEN
                                52          15 00012          BLEQ     7$
                                08 AC        53 D1 00014      CMPL     COUCOD, CTRNAME
                                4C          13 00018          BEQL     7$
                                50          81 3C 0001A      MOVZWL   (CTRPTR)+, COUID
                                53          00 EF 0001D      EXTZV    #0, #12, COUID, COUCOD
                                54          02 C2 00022      SUBL2     #2, CTRLEN
                                50          0C E1 00025      BBC       #12, COUID, 2$
                                55          81 3C 00029      MOVZWL   (CTRPTR)+, ERRMAP
                                54          02 C2 0002C      SUBL2     #2, CTRLEN
                                02          11 0002F          BRB      3$
                                55          D4 00031 2$:      CLRL     ERRMAP
                                6000 8F      50 B3 00033 3$:      BITW     COUID, #24576
                                08          12 00038          BNEQ     4$
                                50 00000000G 8F D0 0003A      MOVL     #LCP$_INTERNAL, R0
                                04          04 00041          RET
                                01          50 02          0D ED 00042 4$:      CMPZV   #13, #2, COUID, #1
                                05          12 00047          BNEQ     5$
                                52          81 9A 00049      MOVZBL   (CTRPTR)+, COUVAL
                                02          C2 11 0004C          BRB      1$
                                02          50 02          0D ED 0004E 5$:      CMPZV   #13, #2, COUID, #2
                                05          12 00053          BNEQ     6$
                                52          81 3C 00055      MOVZWL   (CTRPTR)+, COUVAL
                                03          50 02          B6 11 00058          BRB      1$
                                05          0D ED 0005A 6$:      CMPZV   #13, #2, COUID, #3
                                52          AF 12 0005F          BNEQ     1$
                                08 AC        53 D1 00066 7$:      CMPL     COUCOD, CTRNAME
                                0C          BC D4 0006C          BEQL     8$
                                09          13 0006A          CLRL     @VALUE
                                52          81 D0 00061          MOVL     (CTRPTR)+, COUVAL
                                53          AA 11 00064          BRB      1$
                                54          53 D1 00066          CMPL     COUCOD, CTRNAME
                                55          09 13 0006A          BEQL     8$
                                56          BC D4 0006C          CLRL     @VALUE

```

LATCP
V04-000

LAT Control Program
LCP_FINDCOU Find the specific device counter

J 12
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 101
(28)

		10	BC	D4	0006F	CLRL	@BITMAP	:	3226	
			50	D4	00072	CLRL	STATUS	:	3227	
				04	00074	RET		:		
OC	BC		52	D0	00075	8\$:	MOVL	COUVAL, @VALUE	:	3231
10	BC		55	D0	00079		MOVL	ERRMAP, @BITMAP	:	3232
	50		01	D0	0007D		MOVL	#1, STATUS	:	3233
				04	00080		RET	:	3239	

; Routine Size: 129 bytes, Routine Base: \$CODE\$ + 0E51


```
3131 3240 1 %SBTTL 'LCP_SHOWSERVCOU Show SERVER counters'
3132 3241 1 ROUTINE LCP_SHOWSERVCOU : NOVALUE =
3133 3242 1
3134 3243 1 ++
3135 3244 1 FUNCTIONAL DESCRIPTION:
3136 3245 1
3137 3246 1 Convert and write the all the SERVER counters to sys$output.
3138 3247 1
3139 3248 1 FORMAL PARAMETERS:
3140 3249 1
3141 3250 1 NONE
3142 3251 1
3143 3252 1 IMPLICIT INPUTS:
3144 3253 1
3145 3254 1 counters in LAT driver
3146 3255 1
3147 3256 1 IMPLICIT OUTPUTS:
3148 3257 1
3149 3258 1 NONE
3150 3259 1
3151 3260 1 ROUTINE VALUE:
3152 3261 1 COMPLETION CODES:
3153 3262 1
3154 3263 1 NONE
3155 3264 1
3156 3265 1 SIDE EFFECTS:
3157 3266 1
3158 3267 1 NONE
3159 3268 1
3160 3269 1 --
3161 3270 1
3162 3271 2 BEGIN
3163 3272 2
3164 3273 2 BIND
3165 3274 2 UNKNOWN = DESCRIPTOR ('UNKNOWN')
3166 3275 2 ;
3167 3276 2
3168 3277 2 LITERAL
3169 3278 2 BUFSIZE = 24
3170 3279 2 ;
3171 3280 2
3172 3281 2 LOCAL
3173 3282 2 DSC : BLOCK [DSC$K 2 BLN, BYTE],
3174 3283 2 CTRBUF : VECTOR [BUFSIZE],
3175 3284 2 PTR : REF BLOCK [,BYTE],
3176 3285 2 STATUS
3177 3286 2 ;
3178 3287 2
3179 3288 2
3180 3289 2 CH$FILL (255, GHBS$ _COMM_AREA, AREA);
3181 3290 2
3182 3291 2
3183 3292 2 Get the counters if we can, and if not then signal the error
3184 3293 2
3185 3294 2 IF NOT
3186 3295 2 (STATUS = $CMKRN (ROUTIN = LCPK_GETSERVCTRS) )
3187 3296 2 THEN
```

```

3188      3297 2      SIGNAL (.STATUS)
3189      3298 2      ;
3190      3299 2      :
3191      3300 2      :
3192      3301 2      : Convert the counters for output one at a time
3193      3302 2      :
3194      3303 2      IF .AREA_RET_SIZE EQL 0
3195      3304 2      THEN
3196      3305 2      SIGNAL (LCPS_NOSERVERS)
3197      3306 2      ELSE
3198      3307 2      BEGIN
3199      3308 3      INCR INDX FROM AREA
3200      3309 4      TO AREA + .AREA_RET_SIZE - (LCB_K_LENGTH + GHBSK_NAMELEN + 1)
3201      3310 3      BY LCB_K_LENGTH + GHBSK_NAMELEN + 1
3202      3311 3      DO
3203      3312 4      BEGIN
3204      3313 4      PTR = .INDX;
3205      3314 4      CTRBUF [1] = .PTR [LCB_L_MSG_RCV];
3206      3315 4      CTRBUF [2] = .PTR [LCB_L_MSG_XMT];
3207      3316 4      CTRBUF [3] = .PTR [LCB_L_MSG_REXMT];
3208      3317 4      CTRBUF [4] = .PTR [LCB_W_SEQ_ERR];
3209      3318 4      CTRBUF [5] = .PTR [LCB_B_INV_MSG];
3210      3319 4      CTRBUF [6] = .PTR [LCB_B_INV_SLOT];
3211      3320 4
3212      3321 4      PTR = .PTR + LCB_K_LENGTH;          ! Skip over counters
3213      3322 4
3214      3323 4      IF .PTR [0, 0, 8, 0] EQL 0
3215      3324 4      THEN
3216      3325 4      CTRBUF [0] = UNKNOWN
3217      3326 4      ELSE
3218      3327 5      BEGIN
3219      3328 5      CTRBUF [0] = DSC;
3220      3329 6      $SETDSC (DSC, .PTR [0, 0, 8, 0], PTR [1, 0, 8, 0])
3221      3330 5      END
3222      3331 4      ;
3223      3332 4
3224      3333 4      $SETDSC (OUTDSC, OUTBUFSIZE, OUTBUF);
3225      3334 4
3226      3335 4      STATUS = $FAOL
3227      3336 4      (
3228      3337 4      CTRSTR = SERVCTRSTR,
3229      3338 4      OUTLEN = OUTDSC [DSC$W_LENGTH],
3230      3339 4      OUTBUF = OUTDSC,
3231      3340 4      PRMLST = CTRBUF [0]
3232      3341 4      );
3233      3342 4
3234      3343 4      IF NOT .STATUS THEN SIGNAL (.STATUS);
3235      3344 4
3236      3345 4      STATUS = LIB$PUT_OUTPUT (OUTDSC);
3237      3346 4
3238      3347 4      IF NOT .STATUS THEN SIGNAL (.STATUS)
3239      3348 4
3240      3349 4      END
3241      3350 3      ;
3242      3351 3
3243      3352 3      END
3244      3353 3

```


LATCP
V04-000
: 3245

LAT Control Program
LCP_SHOWSERVCOU Show SERVER counters
3354 1 END;

M 12
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 104
(29)

.PSECT \$SPLITS\$,NOWRT,NOEXE,2

4E 57 4F 4E 4B 4E 55 00E10 P.AFR: .ASCII \UNKNOWN\
00E17 .BLKB 1
00000007 00E18 P.AFQ: .LONG 7
00000000 00E1C .ADDRESS P.AFR
UNKNOWN= P.AFQ

.PSECT \$CODE\$,NOWRT,2

0060	8F	FF	8F	00000000G	00	9E	00002	.WORD	Save R2,R3,R4,R5,R6,R7	: 3241
				57 00000000G	00	9E	00002	MOVAB	LIB\$SIGNAL, R7	:
				56 0000	CF	9E	00009	MOVAB	OUTDSC, R6	:
				5E 98	AE	9E	0000E	MOVAB	-104(SP), SP	:
				6E	00	2C	00012	MOVCS	#0, (SP), #255, #96, AREA	: 3289
				F5FC	C6		0001A			:
					7E	D4	0001D	CLRL	-(SP)	: 3295
				0000V	CF	9F	0001F	PUSHAB	LCPK_GETSERVCTRS	:
				00000000G	00	02	FB	CALLS	#2, SYSSCMKRN	:
				53	50	D0	0002A	MOVL	R0, STATUS	:
				05	53	E8	0002D	BLBS	STATUS, 1\$	:
					53	DD	00030	PUSHL	STATUS	: 3297
				67	01	FB	00032	CALLS	#1, LIB\$SIGNAL	:
				52	FC	A6	D0	MOVL	AREA_RET_SIZE, R2	: 3303
					0A	12	00039	BNEQ	2\$	:
				00000000G	8F	DD	0003B	PUSHL	#LCP\$ NOSERVERS	: 3305
				67	01	FB	00041	CALLS	#1, LIB\$SIGNAL	:
					04		00044	RET		:
				54	F5FC	C6	9E	MOVAB	AREA, R4	: 3308
				55	F5DB	C642	9E	MOVAB	AREA-33[R2], R5	: 3309
					7B	11	00050	BRB	8\$	: 3310
				52	54	D0	00052	MOVL	INDX, PTR	: 3313
04	AE	04	A2	D0	00055			MOVL	4(PTR), CTRBUF+4	: 3314
08	AE		B2	D0	0005A			MOVL	(PTR)+, CTRBUF+8	: 3315
0C	AE	04	A2	D0	0005E			MOVL	4(PTR), CTRBUF+12	: 3316
10	AE	08	A2	3C	00063			MOVZWL	8(PTR), CTRBUF+16	: 3317
14	AE	0A	A2	9A	00068			MOVZBL	10(PTR), CTRBUF+20	: 3318
18	AE	0B	A2	9A	0006D			MOVZBL	11(PTR), CTRBUF+24	: 3319
	52		0C	C0	00072			ADDL2	#12, PTR	: 3321
			62	95	00075			TSTB	(PTR)	: 3323
			07	12	00077			BNEQ	4\$	:
	6E	0000	CF	9E	00079			MOVAB	UNKNOWN, CTRBUF	: 3325
			10	11	0007E			BRB	5\$	:
	6E	60	AE	9E	00080			MOVAB	DSC, CTRBUF	: 3328
		62	AE	B4	00084			CLRW	DSC+2	: 3329
60	AE		62	9B	00087			MOVZBW	(PTR), DSC	:
64	AE	01	A2	9E	0008B			MOVAB	1(R2), DSC+4	:
	66	0400	8F	3C	00090			MOVZWL	#1024, OUTDSC	: 3333
04	A6	08	A6	9E	00095			MOVAB	OUTBUF, OUTDSC+4	:
		4040	8F	BB	0009A			PUSHR	#*M<R6,SP>	: 3341

LATCP
V04-000

LAT Control Program
LCP_SHOWSERVCOU Show SERVER counters

N 12
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 105
(29)

		0000'	56 DD 0009E	PUSHL R6		
			CF 9F 000A0	PUSHAB	SERVCTRSTR	
00000000G	00		04 FB 000A4	CALLS	#4, SYSS\$FAOL	
	53		50 D0 000AB	MOVL	R0, STATUS	
	05		53 E8 000AE	BLBS	STATUS, 6\$	3343
			53 DD 000B1	PUSHL	STATUS	
	67		01 FB 000B3	CALLS	#1, LIB\$SIGNAL	
			56 DD 000B6	PUSHL	R6	3345
00000000G	00		01 FB 000B8	CALLS	#1, LIB\$PUT_OUTPUT	
	53		50 D0 000BF	MOVL	R0, STATUS	
	05		53 E8 000C2	BLBS	STATUS, 7\$	3347
			53 DD 000C5	PUSHL	STATUS	
	67		01 FB 000C7	CALLS	#1, LIB\$SIGNAL	
	54		21 C0 000CA	ADDL2	#33, INDX	
	55		54 D1 000CD	CMPL	INDX, R5	
			80 15 000D0	BLEQ	3\$	
			04 000D2	RET		3354

; Routine Size: 211 bytes, Routine Base: \$CODE\$ + 0ED2


```
3247 3355 1 %SBTTL 'LCP_SHOWHIST Display the history buffer'
3248 3356 1 ROUTINE LCP_SHOWHIST : NOVALUE =
3249 3357 1
3250 3358 1 ++
3251 3359 1 FUNCTIONAL DESCRIPTION:
3252 3360 1
3253 3361 1 Obtain the history buffer from the driver and then display it.
3254 3362 1
3255 3363 1 FORMAL PARAMETERS:
3256 3364 1
3257 3365 1 NONE
3258 3366 1
3259 3367 1 IMPLICIT INPUTS:
3260 3368 1
3261 3369 1 history buffer in the driver.
3262 3370 1
3263 3371 1 IMPLICIT OUTPUTS:
3264 3372 1
3265 3373 1 NONE
3266 3374 1
3267 3375 1 ROUTINE VALUE:
3268 3376 1 COMPLETION CODES:
3269 3377 1
3270 3378 1 errors signalled
3271 3379 1
3272 3380 1 SIDE EFFECTS:
3273 3381 1
3274 3382 1 NONE
3275 3383 1
3276 3384 1 --
3277 3385 1
3278 3386 2 BEGIN
3279 3387 2
3280 3388 2 LOCAL
3281 3389 2 STATUS,
3282 3390 2 CHAR
3283 3391 2 ;
3284 3392 2
3285 3393 2
3286 3394 2 |
3287 3395 2 | Get the history buffer if there is one.
3288 3396 2
3289 3397 2 IF NOT
3290 3398 2 (STATUS = $CMKRNL ( ROUTIN = LCPK_GETHISTORY ) )
3291 3399 2 THEN
3292 3400 2 BEGIN
3293 3401 2 SIGNAL (.STATUS);
3294 3402 2 RETURN;
3295 3403 2 END
3296 3404 2 ;
3297 3405 2
3298 3406 2 |
3299 3407 2 | Now display the history buffer. First the title line and then
3300 3408 2 | the buffer a line at a time.
3301 3409 2
3302 3410 2 LCP_FA00OUTPUT (FAOSTR1, 0);
3303 3411 2 INCR IDX FROM AREA TO AREA + GHBSK_HISTSIZ - 32 BY 32
```

68
66
73
42
20
63
00


```
3310 3417 1 %SBTTL 'LCP_SHOWCHAR Display the LAT driver characteristics'
3311 3418 1 ROUTINE LCP_SHOWCHAR : NOVALUE =
3312 3419 1
3313 3420 1 ++
3314 3421 1 FUNCTIONAL DESCRIPTION:
3315 3422 1
3316 3423 1 Obtain the common data area from the driver and then display
3317 3424 1 interesting tidbits.
3318 3425 1
3319 3426 1 FORMAL PARAMETERS:
3320 3427 1
3321 3428 1 NONE
3322 3429 1
3323 3430 1 IMPLICIT INPUTS:
3324 3431 1
3325 3432 1 common data area in the driver.
3326 3433 1
3327 3434 1 IMPLICIT OUTPUTS:
3328 3435 1
3329 3436 1 NONE
3330 3437 1
3331 3438 1 ROUTINE VALUE:
3332 3439 1 COMPLETION CODES:
3333 3440 1
3334 3441 1 errors signalled
3335 3442 1
3336 3443 1 SIDE EFFECTS:
3337 3444 1
3338 3445 1 NONE
3339 3446 1
3340 3447 1 --
3341 3448 1
3342 3449 2 BEGIN
3343 3450 2
3344 3451 2 LOCAL
3345 3452 2 STATUS,
3346 3453 2 PTR,
3347 3454 2 NUM_SERVICES
3348 3455 2 ;
3349 3456 2
3350 3457 2 LITERAL
3351 3458 2 STRSIZE = 512, ! Size of input and output strings
3352 3459 2 PRMSIZE = 24 ! Size of parameter list in LONGs
3353 3460 2 ;
3354 3461 2
3355 3462 2 LOCAL
3356 3463 2 DSC : BLOCK [DSC$K 2 BLN, BYTE],
3357 3464 2 BUF : VECTOR [STRSIZE, BYTE],
3358 3465 2 STR : REF VECTOR [, BYTE],
3359 3466 2 VER_ECO : VECTOR [2, BYTE],
3360 3467 2 P : VECTOR [PRMSIZE]
3361 3468 2 ;
3362 3469 2
3363 3470 2 !
3364 3471 2 Get the COMMON AREA in the driver and setup the descriptors.
3365 3472 2
3366 3473 2 IF NOT
```

```

: 3367      3474 3      (STATUS = $CMKRN (ROUTIN = LCPK_GETCOMMON) )
: 3368      3475 2      THEN
: 3369      3476 2      SIGNAL (.STATUS)
: 3370      3477 2      ;
: 3371      3478 2      |
: 3372      3479 2      | Initialize descriptors
: 3373      3480 2      |
: 3374      3481 2      $SETDSC (IDDSC, GHBSK_IDLEN, ID);
: 3375      3482 2      $SETDSC (NODE_IDDSC, GHBSK_IDLEN, NODE_ID);
: 3376      3483 2      $SETDSC (SERVDSC, GHBSK_NAMELEN, SERV);
: 3377      3484 2      $SETDSC (NAMEDSC, GHBSK_NAMELEN, NAME);
: 3378      3485 2      $SETDSC (NAMEDSC, GHBSK_NAMELEN, NAME);
: 3379      3486 2      STR = .OUTDSC [DSC$A_POINTER];
: 3380      3487 2      STR = .OUTDSC [DSC$A_POINTER];
: 3381      3488 2      STR = .OUTDSC [DSC$A_POINTER];
: 3382      3489 2      STR = .OUTDSC [DSC$A_POINTER];
: 3383      3490 2      STR = .OUTDSC [DSC$A_POINTER];
: 3384      3491 2      STR = .OUTDSC [DSC$A_POINTER];
: 3385      3492 2      |
: 3386      3493 2      | Pick up group codes, timer and skip host node status
: 3387      3494 2      |
: 3388      3495 2      MCASTIMR = .STR [0];
: 3389      3496 2      PTR = CH$MOVE (CH$RCHAR (STR [2]), STR [3], GROUPS);
: 3390      3497 2      PTR = CH$MOVE (CH$RCHAR (STR [2]), STR [3], GROUPS);
: 3391      3498 2      PTR = CH$MOVE (CH$RCHAR (STR [2]), STR [3], GROUPS);
: 3392      3499 2      |
: 3393      3500 2      | Skip the group codes and host node name.
: 3394      3501 2      STR = .STR [2] + STR [3];          ! Skip group codes
: 3395      3502 2      STR = .STR [0] + STR [1];          ! Skip host node name
: 3396      3503 2      STR = .STR [0] + STR [1];          ! Skip host node name
: 3397      3504 2      STR = .STR [0] + STR [1];          ! Skip host node name
: 3398      3505 2      |
: 3399      3506 2      | Get host node id, then skip over it.
: 3400      3507 2      |
: 3401      3508 2      NODE_IDDSC [DSC$W_LENGTH] = .STR [0];
: 3402      3509 2      PTR = CH$MOVE (.NODE_IDDSC [DSC$W_LENGTH],
: 3403      3510 2      STR [1], .NODE_IDDSC [DSC$A_POINTER]);
: 3404      3511 2      STR = .STR [0] + STR [1];          ! Skip host node id.
: 3405      3512 2      STR = .STR [0] + STR [1];          ! Skip host node id.
: 3406      3513 2      |
: 3407      3514 2      | Get number of services
: 3408      3515 2      |
: 3409      3516 2      NUM_SERVICES = .STR [0];
: 3410      3517 2      NUM_SERVICES = .STR [0];
: 3411      3518 2      |
: 3412      3519 2      | Get the 1st service name and descriptor
: 3413      3520 2      |
: 3414      3521 2      NAMEDSC [DSC$W_LENGTH] = .STR [2];
: 3415      3522 2      PTR = CH$MOVE (.NAMEDSC [DSC$W_LENGTH], STR [3], .NAMEDSC [DSC$A_POINTER]);
: 3416      3523 2      PTR = CH$MOVE (.NAMEDSC [DSC$W_LENGTH], STR [3], .NAMEDSC [DSC$A_POINTER]);
: 3417      3524 2      STR = .STR [2] + STR [3];          ! Skip 1st service name
: 3418      3525 2      |
: 3419      3526 2      |
: 3420      3527 2      |
: 3421      3528 2      |
: 3422      3529 2      | If we have a second service name, then report the second service here
: 3423      3530 2      |

```



```
3424 3531 2 IF .NUM_SERVICES LEQ 1
3425 3532 2 THEN
3426 3533 2     SERVDSC [DSC$W_LENGTH] = 0
3427 3534 2 ELSE
3428 3535 2     BEGIN
3429 3536 2         STR = .STR [0] + STR [1];           ! Skip 1st service id
3430 3537 2         STR = .STR + 1;                   ! Skip 2nd rating
3431 3538 2         SERVDSC [DSC$W_LENGTH] = .STR [0];
3432 3539 2         PTR = CH$MOVE (.SERVDSC [DSC$W_LENGTH], STR [1],
3433 3540 2             .SERVDSC [DSC$A_POINTER])
3434 3541 2     END
3435 3542 2 ;
3436 3543 2
3437 3544 2
3438 3545 2     Now display the relevant parts of the common area.
3439 3546 2
3440 3547 2
3441 3548 2
3442 3549 2     Convert the group codes to a nice printable ascii string
3443 3550 2
3444 3551 2 LCP_CVTGROUPS ();
3445 3552 2
3446 3553 2
3447 3554 2     Set up the parameter list and report the current characteristics
3448 3555 2
3449 3556 2 P[0] = NAMEDSC;
3450 3557 2 P[1] = SERVDSC;
3451 3558 2 P[2] = NODE_IDDSC;
3452 3559 2 P[3] = IDDSC;
3453 3560 2 P[4] = GRPDSC;
3454 3561 2 P[5] = .MCASTIMR;
3455 3562 2 P[6] = .AREA [GHBSB_LATVERSION];
3456 3563 2 P[7] = .AREA [GHBSB_LATECO];
3457 3564 2
3458 3565 2 IF .AREA [GHBSL_TIM_ACT] NEQ 0
3459 3566 2 THEN
3460 3567 2     P[8] = ACTIVE
3461 3568 2 ELSE
3462 3569 2     P[8] = INACTIVE;
3463 3570 2
3464 3571 2 $SETDSC (DSC, STRSIZE, BUF);
3465 3572 2
3466 3573 2 $FAOL
3467 3574 2 (
3468 3575 2     CTRSTR = %ASCII
3469 3576 2     %STRING
3470 3577 2     (
3471 3578 2         '!/LCP Characteristics!//',
3472 3579 2         'Node name = !18<\!AS\!> Service name = !18<\!AS\!>!//',
3473 3580 2         'Node Identification = \!AS\!//',
3474 3581 2         'Service Identification = \!AS\!//',
3475 3582 2         'Groups = !AS!//',
3476 3583 2         'Multicast timer = !UB seconds!//',
3477 3584 2         'LAT Version = !16<!UB.!UB!> LAT Protocol is !AS!//',
3478 3585 2     )
3479 3586 2 )
3480 3587 2 OUTBUF = DSC,
    OUTLEN = DSC,
```

LATCP
V04-000

LAT Control Program
LCP_SHOWCHAR Display the LAT driver characteri

G 13
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 111
(31)

```
: 3481      P 3588 2      PRMLST = P[0]
: 3482      3589 2      );
: 3483      3590 2
: 3484      3591 2      LIB$PUT_OUTPUT (DSC)
: 3485      3592 2
: 3486      3593 1      END;
```

```
72 65 74 63 61 72 61 68 43 20 50 43 4C 2F 21 00E20 P.AFT: .PSECT $SPLITS,NOWRT,NOEXE,2
20 65 64 6F 4E 2F 21 2F 21 73 63 69 74 73 69 00E2F .ASCII \!/LCP Characteristics!//Node name = !18\
76 72 65 53 20 20 20 3E 21 5C 53 41 21 5C 3C 00E3E .ASCII \<\<92>\!AS\<92>\!> Service name = !1\
65 64 6F 4E 2F 21 3E 21 5C 53 41 21 5C 3C 00E48 .ASCII \8<\<92>\!AS\<92>\!>!/Node Identificati\
72 65 53 2F 21 3E 21 5C 53 41 21 5C 3C 00E57 .ASCII \on = \<92>\!AS\<92>\!>/Service Identifi\
21 5C 53 41 21 5C 20 3D 20 73 70 75 6F 72 47 2F 00E64 .ASCII \cation = \<92>\!AS\<92>\!>/Groups = !AS\
6D 69 74 20 74 73 61 63 69 74 6C 75 4D 2F 21 00E73 .ASCII \!/Multicast timer = !UB seconds!/LAT Ver\
64 6E 6F 63 65 73 20 42 55 21 20 3D 20 72 65 00E80 .ASCII \sion = !16<!UB.!UB!> LAT Protocol is !\
2E 42 55 21 3C 36 31 21 20 3D 20 6E 6F 69 73 00E8F .ASCII \AS!/\
6F 72 50 20 54 41 4C 20 20 20 3E 21 42 55 21 00E9C .LONG 17694956
010E00EC 00F0C P.AFS: .ADDRESS P.AFT
00000000 00F10
```

```
.PSECT $CODE$,NOWRT,2
03FC 00000 LCP_SHOWCHAR:
59 0000' CF 9E 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8,R9 : 3418
5E FD98 CE 9E 00007 MOVAB SERVDSC, R9
7E D4 0000C MOVAB -616(SP), SP : 3474
0000V CF 9F 0000E CLRL -(SP)
00000000G 00 02 FB 00012 PUSHAB LCPK_GETCOMMON
09 50 E8 00019 CALLS #2, SYSSCMKRNL
00000000G 00 50 DD 0001C BLBS STATUS, 1$ : 3476
48 A9 40 8F 9A 00025 1$: CALLS #1, LIB$SIGNAL : 3482
4C A9 08 A9 9E 0002A MOVZBL #64, IDDSC : 3484
0090 C9 40 8F 9A 0002F MOVAB ID, IDDSC+4
0094 C9 50 A9 9E 00035 MOVZBL #64, NODE_ID
69 10 D0 0003B MOVAB NODE_ID, NODE_ID+4 : 3486
04 A9 F0 A9 9E 0003E MOVL #16, SERVDSC : 3488
E8 A9 10 D0 00043 MOVAB SERV, SERVDSC+4
EC A9 D8 A9 9E 00047 MOVL #16, NAMEDSC : 3490
00E0 56 0AF4 C9 D0 0004C MOVAB NAME, NAMEDSC+4
C9 66 9A 00051 MOVL OUTDSC+4, STR : 3495
50 02 A6 9A 00056 MOVZBL 2(STR), R0 : 3496
```


00B0	C9	57	03	A6	9E	0005A	MOVAB	3(STR), R7	:	
		67		50	28	0005E	MOVAB	R0, (R7), GROUPS	:	
		56	02	A6	9A	00064	MOVZBL	2(STR), STR	:	3501
		56		57	C0	00068	ADDL2	R7, STR	:	
		50		66	9A	0006B	MOVZBL	(STR), R0	:	3502
		50		56	C0	0006E	ADDL2	STR, R0	:	
		56	01	A0	9E	00071	MOVAB	1(R0), STR	:	
	0090	C9		66	9B	00075	MOVZBW	(STR), NODE_IDDSC	:	3507
		57	01	A6	9E	0007A	MOVAB	1(STR), R7	:	3509
0094	D9	67	0090	C9	28	0007E	MOVAB	NODE_IDDSC, (R7), @NODE_IDDSC+4	:	
		56		66	9A	00086	MOVZBL	(STR), STR	:	3511
		56		57	C0	00089	ADDL2	R7, STR	:	
		58		66	9A	0008C	MOVZBL	(STR), NUM_SERVICES	:	3516
	E8	A9	02	A6	9B	0008F	MOVZBW	2(STR), NAMEDSC	:	3521
		57	03	A6	9E	00094	MOVAB	3(STR), R7	:	3522
EC	B9	67	E8	A9	28	00098	MOVAB	NAMEDSC, (R7), @NAMEDSC+4	:	
		56	02	A6	9A	0009E	MOVZBL	2(STR), STR	:	3524
		56		57	C0	000A2	ADDL2	R7, STR	:	
	48	A9		66	9B	000A5	MOVZBW	(STR), IDDSC	:	3525
		57	01	A6	9E	000A9	MOVAB	1(STR), R7	:	3526
4C	B9	67	48	A9	28	000AD	MOVAB	IDDSC, (R7), @IDDSC+4	:	
		01		58	D1	000B3	CMPL	NUM_SERVICES, #1	:	3531
				04	14	000B6	BGTR	2\$	:	
				69	B4	000B8	CLRW	SERVDSC	:	3533
				11	11	000BA	BRB	3\$	:	
		56		66	9A	000BC	MOVZBL	(STR), STR	:	3536
		56		57	C0	000BF	ADDL2	R7, STR	:	
				56	D6	000C2	INCL	STR	:	3537
		69		66	9B	000C4	MOVZBW	(STR), SERVDSC	:	3538
04	B9	A6		69	28	000C7	MOVAB	SERVDSC, 1(STR), @SERVDSC+4	:	3540
		CF	01	00	FB	000CD	CALLS	#0, LCP_CVTGROUPS	:	3551
		6E	E8	A9	9E	000D2	MOVAB	NAMEDSC, P	:	3556
	04	AE		69	9E	000D6	MOVAB	SERVDSC, P+4	:	3557
	03	AE	0090	C9	9E	000DA	MOVAB	NODE_IDDSC, P+8	:	3558
	0C	AE	48	A9	9E	000E0	MOVAB	IDDSC, P+12	:	3559
	10	AE	0EF8	C9	9E	000E5	MOVAB	GRPDSC, P+16	:	3560
	14	AE	00E0	C9	D0	000EB	MOVL	MCASTIMR, P+20	:	3561
	18	AE	00EE	C9	9A	000F1	MOVZBL	AREA+2, P+24	:	3562
	1C	AE	00EF	C9	9A	000F7	MOVZBL	AREA+3, P+28	:	3563
			0104	C9	D5	000FD	TSTL	AREA+24	:	3565
				08	13	00101	BEQL	4\$	:	
	20	AE	0000'	CF	9E	00103	MOVAB	ACTIVE, P+32	:	3567
				06	11	00109	BRB	5\$	:	
	20	AE	0000'	CF	9E	0010B	MOVAB	INACTIVE, P+32	:	3569
	F8	AD	0200	8F	3C	00111	MOVZWL	#512, DSC	:	3571
	FC	AD	60	AE	9E	00117	MOVAB	BUF, DSC+4	:	
				5E	DD	0011C	PUSHL	SP	:	3589
				F8	AD	9F	PUSHAB	DSC	:	
				F8	AD	9F	PUSHAB	DSC	:	
			0000'	CF	9F	00124	PUSHAB	P.AFS	:	
	00000000G	00		04	FB	00128	CALLS	#4, SYSS\$FAOL	:	3591
			F8	AD	9F	0012F	PUSHAB	DSC	:	
	00000000G	00		01	FB	00132	CALLS	#1, LIB\$PUT_OUTPUT	:	3593
				04	00	00139	RET		:	

; Routine Size: 314 bytes, Routine Base: \$CODE\$ + 0FEC

LATCP
V04-000

LAT Control Program
LCP_SHOWCHAR Display the LAT driver characteri

I 13
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 113
(31)

LAT
V04


```
3488 3594 1 %SBTTL 'LCP_CVTGROUPS Convert the group codes to an ascii printable string'
3489 3595 1 ROUTINE LCP_CVTGROUPS : NOVALUE =
3490 3596 1
3491 3597 1 ++
3492 3598 1 FUNCTIONAL DESCRIPTION:
3493 3599 1
3494 3600 1 Convert the group codes in thr GROUPS vector to an ascii output
3495 3601 1 string, suitable for printing.
3496 3602 1
3497 3603 1 FORMAL PARAMETERS:
3498 3604 1
3499 3605 1 NONE
3500 3606 1
3501 3607 1 IMPLICIT INPUTS:
3502 3608 1
3503 3609 1 GROUP vector area set up with group codes
3504 3610 1
3505 3611 1 IMPLICIT OUTPUTS:
3506 3612 1
3507 3613 1 NONE
3508 3614 1
3509 3615 1 ROUTINE VALUE:
3510 3616 1 COMPLETION CODES:
3511 3617 1
3512 3618 1 errors signalled
3513 3619 1
3514 3620 1 SIDE EFFECTS:
3515 3621 1
3516 3622 1 NONE
3517 3623 1
3518 3624 1 --
3519 3625 1
3520 3626 2 BEGIN
3521 3627 2
3522 3628 2 LOCAL
3523 3629 2 STATUS,
3524 3630 2 BITPOS, ! Bit position within longword
3525 3631 2 SIZE, ! Length of bit stream to scan
3526 3632 2 BYTE_OFFSET, ! Offset in group code string
3527 3633 2 PTR, ! Pointer into output string
3528 3634 2 LEFT_PAREN,
3529 3635 2 RIGHT_PAREN,
3530 3636 2 COMMA,
3531 3637 2 INDEX
3532 3638 2 ;
3533 3639 2
3534 3640 2 LITERAL
3535 3641 2 STRSIZE = 512, ! Size of input and output strings
3536 3642 2 BITSIZE = 32, ! Maximum number of bits to scan
3537 3643 2 CVTSTRSIZE = 3 ! Size of output conversion string
3538 3644 2 ;
3539 3645 2
3540 3646 2 LOCAL
3541 3647 2 CVTDSC : BLOCK [DSC$K_Z_BLN, BYTE],
3542 3648 2 CVTSTR : VECTOR [CVTSTRSIZE, BYTE],
3543 3649 2 GROUP : REF BLOCK [,BYTE]
3544 3650 2 ;
```

```
3545 3651 2
3546 3652 2 LEFT_PAREN = %C('('; RIGHT_PAREN = %C(')'; COMMA = %C(',');
3547 3653 2
3548 3654 2
3549 3655 2
3550 3656 2
3551 3657 2
3552 3658 2
3553 3659 2
3554 3660 2
3555 3661 2
3556 3662 2
3557 3663 2
3558 3664 2
3559 3665 2
3560 3666 2
3561 3667 2
3562 3668 2
3563 3669 2
3564 3670 2
3565 3671 2
3566 3672 2
3567 3673 2
3568 3674 2
3569 3675 2
3570 3676 2
3571 3677 2
3572 3678 2
3573 3679 2
3574 3680 2
3575 3681 2
3576 3682 2
3577 3683 2
3578 3684 2
3579 3685 2
3580 3686 2
3581 3687 4
3582 3688 4
3583 3689 4
3584 3690 4
3585 3691 4
3586 3692 5
3587 3693 5
3588 3694 5
3589 3695 5
3590 3696 5
3591 3697 5
3592 3698 5
3593 3699 5
3594 3700 5
3595 3701 5
3596 3702 5
3597 3703 5
3598 3704 5
3599 3705 5
3600 3706 5
3601 3707 5

      Initialize the group codes input string
$SETDSC (GRPDSC, STRSIZE, GRPBUF);
PTR = .GRPDSC [DSC$A_POINTER];
PTR = CH$MOVE(1, LEFT_PAREN, .PTR);

      Initialize the conversion output string
$SETDSC (CVDSC, CVTSTRSIZE, CVTSTR);

      Set up the Group codes string

      BYTE_OFFSET = 0;
      GROUP = GROUPS;
      ! Start from beginning
      ! Copy the group code pointer

      Sear the group codes one longword at a time, note that there are
      32 longwords that we must search. We will use the FFS library routine
      to find all corresponding bits that are set.

      WHILE .BYTE_OFFSET LSS 32
      DO
      BEGIN
      BITPOS = 0;
      SIZE = BITSIZE;
      STATUS = TRUE;
      WHILE .STATUS AND .BITPOS LSS BITSIZE
      DO
      BEGIN
      STATUS = LIB$FFS(BITPOS, SIZE, GROUP[.BYTE_OFFSET,0,0,0],
      BITPOS);
      IF .STATUS
      THEN
      BEGIN
      ! Convert the bit number + offset to a binary value,
      ! and strip off leading spaces from converted string.
      STATUS = OT$CVT L TI (%REF((.BYTE_OFFSET*8)+.BITPOS), CVDSC);
      INDEX = LIB$SKPCT(%ASCID %STRING(' '), CVDSC) - 1;
      ! Move the converted string to the output buffer and
      ! add a ',' to the end.
      PTR = CH$MOVE (.CVDSC [DSC$W_LENGTH] - .INDEX,
      .CVDSC [DSC$A_POINTER] + .INDEX, .PTR);
      PTR = CH$MOVE(1, COMMA, .PTR);
      BITPOS = .BITPOS + 1;
      SIZE = .SIZE - .BITPOS
```


PC	OP	OP2	OP3	OP4	OP5	OP6	OP7	OP8	OP9	OP10	OP11	OP12	OP13	OP14	OP15	OP16	OP17	OP18	OP19	OP20	OP21	OP22	OP23	OP24	OP25	OP26	OP27	OP28	OP29	OP30	OP31	OP32	OP33	OP34	OP35	OP36	OP37	OP38	OP39	OP40	OP41	OP42	OP43	OP44	OP45	OP46	OP47	OP48	OP49	OP50	OP51	OP52	OP53	OP54	OP55	OP56	OP57	OP58	OP59	OP60	OP61	OP62	OP63	OP64	OP65	OP66	OP67	OP68	OP69	OP70	OP71	OP72	OP73	OP74	OP75	OP76	OP77	OP78	OP79	OP80	OP81	OP82	OP83	OP84	OP85	OP86	OP87	OP88	OP89	OP90	OP91	OP92	OP93	OP94	OP95	OP96	OP97	OP98	OP99	OP100	OP101	OP102	OP103	OP104	OP105	OP106	OP107	OP108	OP109	OP110	OP111	OP112	OP113	OP114	OP115	OP116	OP117	OP118	OP119	OP120	OP121	OP122	OP123	OP124	OP125	OP126	OP127	OP128	OP129	OP130	OP131	OP132	OP133	OP134	OP135	OP136	OP137	OP138	OP139	OP140	OP141	OP142	OP143	OP144	OP145	OP146	OP147	OP148	OP149	OP150	OP151	OP152	OP153	OP154	OP155	OP156	OP157	OP158	OP159	OP160	OP161	OP162	OP163	OP164	OP165	OP166	OP167	OP168	OP169	OP170	OP171	OP172	OP173	OP174	OP175	OP176	OP177	OP178	OP179	OP180	OP181	OP182	OP183	OP184	OP185	OP186	OP187	OP188	OP189	OP190	OP191	OP192	OP193	OP194	OP195	OP196	OP197	OP198	OP199	OP200	OP201	OP202	OP203	OP204	OP205	OP206	OP207	OP208	OP209	OP210	OP211	OP212	OP213	OP214	OP215	OP216	OP217	OP218	OP219	OP220	OP221	OP222	OP223	OP224	OP225	OP226	OP227	OP228	OP229	OP230	OP231	OP232	OP233	OP234	OP235	OP236	OP237	OP238	OP239	OP240	OP241	OP242	OP243	OP244	OP245	OP246	OP247	OP248	OP249	OP250	OP251	OP252	OP253	OP254	OP255	OP256	OP257	OP258	OP259	OP260	OP261	OP262	OP263	OP264	OP265	OP266	OP267	OP268	OP269	OP270	OP271	OP272	OP273	OP274	OP275	OP276	OP277	OP278	OP279	OP280	OP281	OP282	OP283	OP284	OP285	OP286	OP287	OP288	OP289	OP290	OP291	OP292	OP293	OP294	OP295	OP296	OP297	OP298	OP299	OP300	OP301	OP302	OP303	OP304	OP305	OP306	OP307	OP308	OP309	OP310	OP311	OP312	OP313	OP314	OP315	OP316	OP317	OP318	OP319	OP320	OP321	OP322	OP323	OP324	OP325	OP326	OP327	OP328	OP329	OP330	OP331	OP332	OP333	OP334	OP335	OP336	OP337	OP338	OP339	OP340	OP341	OP342	OP343	OP344	OP345	OP346	OP347	OP348	OP349	OP350	OP351	OP352	OP353	OP354	OP355	OP356	OP357	OP358	OP359	OP360	OP361	OP362	OP363	OP364	OP365	OP366	OP367	OP368	OP369	OP370	OP371	OP372	OP373	OP374	OP375	OP376	OP377	OP378	OP379	OP380	OP381	OP382	OP383	OP384	OP385	OP386	OP387	OP388	OP389	OP390	OP391	OP392	OP393	OP394	OP395	OP396	OP397	OP398	OP399	OP400	OP401	OP402	OP403	OP404	OP405	OP406	OP407	OP408	OP409	OP410	OP411	OP412	OP413	OP414	OP415	OP416	OP417	OP418	OP419
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

LATCP
V04-000

LAT Control Program
LCP_CVTGROUPS Convert the group codes to an as

M 13
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 117
(32)

		14	AE	9F	00053	PUSHAB	SIZE	:	
		1C	AE	9F	00056	PUSHAB	BITPOS	:	
00000000G	00		04	FB	00059	CALLS	#4, LIB\$FFS	:	
	5A		50	D0	00060	MOVL	R0, STATUS	:	
	42		5A	E9	00063	BLBC	STATUS, 3\$	:	3690
		14	AE	9F	00066	PUSHAB	CVTDSC	:	3697
04	AE	14	BE47	7E	00069	MOVAQ	@BITPOS[BYTE_OFFSET], 4(SP)	:	
		04	AE	9F	0006F	PUSHAB	4(SP)	:	
00000000G	00		02	FB	00072	CALLS	#2, OTS\$CVT_L_TI	:	
	5A		50	D0	00079	MOVL	R0, STATUS	:	
		14	AE	9F	0007C	PUSHAB	CVTDSC	:	3698
		0000'	CF	9F	0007F	PUSHAB	P,AFU	:	
00000000G	00		02	FB	00083	CALLS	#2, LIB\$SKPC	:	
	58		FF	A0	0008A	MOVAB	-1(R0), INDEX	:	
	50		14	AE	3C	MOVZWL	CVTDSC, R0	:	3703
	50		58	C2	00092	SUBL2	INDEX, R0	:	
63	18	BE48	50	28	00095	MOVCL	R0, @CVTDSC+4[INDEX], (PTR)	:	3704
	83		5B	90	0009B	MOVB	COMMA, (PTR)+	:	3705
		10	AE	D6	0009E	INCL	BITPOS	:	3706
0C	AE	10	AE	C2	000A1	SUBL2	BITPOS, SIZE	:	3707
			9D	11	000A6	BRB	2\$	:	3685
	57		04	C0	000A8	ADDL2	#4, BYTE_OFFSET	:	3712
			88	11	000AB	BRB	1\$	:	
	73	04	AE	90	000AD	MOVB	RIGHT_PAREN, -(PTR)	:	3720
			53	D6	000B1	INCL	PTR	:	
0000'	CF	53	0000'	CF	A3	SUBW3	GRPDSC+4, PTR, GRPDSC	:	3725
				04	000BB	RET		:	3727

; Routine Size: 188 bytes, Routine Base: \$CODE\$ + 1126

LATCP
V04-000

LAT Control Program
LCP_SHOWSERVERS Display the LAT Servers

N 13
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMMASTER:[LAT.SRC]LATCP.B32;1 Page 118
(33)

```

3623 3728 1 %SBTTL 'LCP_SHOWSERVERS Display the LAT Servers'
3624 3729 1 ROUTINE LCP_SHOWSERVERS : NOVALUE =
3625 3730 1
3626 3731 1 ++
3627 3732 1 FUNCTIONAL DESCRIPTION:
3628 3733 1
3629 3734 1 Show the LAT servers (by name) and number of active sessions.
3630 3735 1 interesting tidbits.
3631 3736 1
3632 3737 1 FORMAL PARAMETERS:
3633 3738 1
3634 3739 1 NONE
3635 3740 1
3636 3741 1 IMPLICIT INPUTS:
3637 3742 1
3638 3743 1 common data area in the driver.
3639 3744 1
3640 3745 1 IMPLICIT OUTPUTS:
3641 3746 1
3642 3747 1 NONE
3643 3748 1
3644 3749 1 ROUTINE VALUE:
3645 3750 1 COMPLETION CODES:
3646 3751 1
3647 3752 1 errors signalled
3648 3753 1
3649 3754 1 SIDE EFFECTS:
3650 3755 1
3651 3756 1 NONE
3652 3757 1
3653 3758 1 --
3654 3759 1
3655 3760 2 BEGIN
3656 3761 2
3657 3762 2 BIND
3658 3763 2 UNKNOWN = DESCRIPTOR ('UNKNOWN')
3659 3764 2 ;
3660 3765 2
3661 3766 2 LITERAL
3662 3767 2 BUFSIZE = 24
3663 3768 2 ;
3664 3769 2
3665 3770 2 LOCAL
3666 3771 2 NAME DSC : BLOCK [DSC$K Z BLN, BYTE],
3667 3772 2 PARAM_BUF : VECTOR [BUFSIZE],
3668 3773 2 PTR : REF BLOCK [,BYTE],
3669 3774 2 STATUS
3670 3775 2 ;
3671 3776 2
3672 3777 2
3673 3778 2 CH$FILL (255, GHBS$COMM_AREA, AREA);
3674 3779 2
3675 3780 2
3676 3781 2 ! Get the counters if we can, and if not then signal the error
3677 3782 2
3678 3783 2 IF NOT
3679 3784 3 (STATUS = $CMKRNL (ROUTIN = LCPK_GETSERVERS) )
```

```
: 3680      3785  2      THEN
: 3681      3786  2      SIGNAL (.STATUS)
: 3682      3787  2      ;
: 3683      3788  2      :
: 3684      3789  2      :
: 3685      3790  2      Convert the server info for output one at a time
: 3686      3791  2      :
: 3687      3792  2      Display:
: 3688      3793  2      :
: 3689      3794  2      Server name
: 3690      3795  2      Server state
: 3691      3796  2      Number of users
: 3692      3797  2      Ethernet address
: 3693      3798  2      :
: 3694      3799  2      IF .AREA_RET_SIZE EQL 0
: 3695      3800  2      THEN
: 3696      3801  2      SIGNAL (LCP$_NOSERVERS)
: 3697      3802  2      ELSE
: 3698      3803  3      BEGIN
: 3699      3804  3      INCR INDX FROM AREA
: 3700      3805  4      TO AREA + .AREA_RET_SIZE - (GHB$_NAMELEN + 9)
: 3701      3806  3      BY GHB$_NAMELEN + 9
: 3702      3807  3      DO
: 3703      3808  4      BEGIN
: 3704      3809  4      PTR = .INDX;
: 3705      3810  4      :
: 3706      3811  4      : Get name descriptor
: 3707      3812  4      :
: 3708      3813  4      IF .PTR [0, 0, 8, 0] EQL 0
: 3709      3814  4      THEN
: 3710      3815  4      PARAM_BUF [0] = UNKNOWN
: 3711      3816  4      ELSE
: 3712      3817  4      BEGIN
: 3713      3818  5      PARAM_BUF [0] = NAME_DSC;
: 3714      3819  5      $SETDSC (NAME_DSC, .PTR [0, 0, 8, 0], PTR [1, 0, 8, 0])
: 3715      3820  6      END
: 3716      3821  5      :
: 3717      3822  4      :
: 3718      3823  4      :
: 3719      3824  4      : Skip over server name
: 3720      3825  4      :
: 3721      3826  4      PTR = .PTR + GHB$_NAMELEN + 1;
: 3722      3827  4      :
: 3723      3828  4      : Get the Ethernet address, 6 bytes (1-6)
: 3724      3829  4      :
: 3725      3830  4      PARAM_BUF [1] = .PTR [1, 0, 8, 0];
: 3726      3831  4      PARAM_BUF [2] = .PTR [2, 0, 8, 0];
: 3727      3832  4      PARAM_BUF [3] = .PTR [3, 0, 8, 0];
: 3728      3833  4      PARAM_BUF [4] = .PTR [4, 0, 8, 0];
: 3729      3834  4      PARAM_BUF [5] = .PTR [5, 0, 8, 0];
: 3730      3835  4      PARAM_BUF [6] = .PTR [6, 0, 8, 0];
: 3731      3836  4      :
: 3732      3837  4      PARAM_BUF [8] = .PTR [0, 0, 8, 0];
: 3733      3838  4      ! Get the # of users
: 3734      3839  4      :
: 3735      3840  4      : Display current state
: 3736      3841  4      :
```



```

: 3737      3842  4      !
: 3738      3843  4      IF .PTR [7, 0, 8, 0]
: 3739      3844  4      THEN
: 3740      3845  4          PARAM_BUF [7] = ACTIVE
: 3741      3846  4      ELSE
: 3742      3847  4          PARAM_BUF [7] = INACTIVE
: 3743      3848  4      ;
: 3744      3849  4
: 3745      3850  4      $SETDSC (OUTDSC, OUTBUFSIZE, OUTBUF);
: 3746      3851  4
: 3747      P 3852  4      STATUS = $FAOL
: 3748      P 3853  4      (
: 3749      L 3854  4          CTRSTR = %ASCID
: 3750      L 3855  4          %STRING
: 3751      L 3856  4          (
: 3752      L 3857  4              !/LCP Server Characteristics for !AS!//',
: 3753      L 3858  4              'Ethernet address = !XB-!XB-!XB-!XB-!XB-!XB !/',
: 3754      L 3859  4              'Server is !18<!AS!>      Active users = !18<!UB!>!/'
: 3755      P 3860  4          )
: 3756      P 3861  4          OUTLEN = OUTDSC [DSC$W_LENGTH],
: 3757      P 3862  4          OUTBUF = OUTDSC,
: 3758      P 3863  4          PRMLST = PARAM_BUF [0]
: 3759      3864  4      );
: 3760      3865  4
: 3761      3866  4      IF NOT .STATUS THEN SIGNAL (.STATUS);
: 3762      3867  4
: 3763      3868  4      STATUS = LIB$PUT_OUTPUT (OUTDSC);
: 3764      3869  4
: 3765      3870  4      IF NOT .STATUS THEN SIGNAL (.STATUS)
: 3766      3871  4
: 3767      3872  4      END
: 3768      3873  3      ;
: 3769      3874  3
: 3770      3875  3      END
: 3771      3876  3
: 3772      3877  3
: 3773      3878  1      END;
```

```

                                .PSECT $SPLITS,NOWRT,NOEXE,2
                                4E 57 4F 4E 4B 4E 55 00F20 P.AFX: .ASCII \UNKNOWN\
                                00000007 00F27 .BLKB 1
                                00000000 0CF28 P.AFW: .LONG 7
                                00000000 00F2C P.AFZ: .ADDRESS P.AFX
                                00F30 .ASCII \!/LCP Server Characteristics for !AS!//\
                                00F3F
                                00F4E
                                00F58 .ASCII \Ethernet address = !XB-!XB-!XB-!XB-!XB-!\
                                00F67
                                00F76
                                00F80 .ASCII \XB !/Server is !18<!AS!>      Active users\
                                00F8F
                                00F9E
                                00FA8 .ASCII \ = !18<!UB!>!/\/<0><0>
                                00FB7
```


010E0086 00FB8 P.AFY: .LONG 17694854
00000000 00FBC .ADDRESS P.AFZ

UNKNOWN=

P.AFW

.PSECT \$CODE\$,NOWRT,2

00FC 00000 LCP_SHOWSERVERS:

0060	8F	FF	8F	57	00000000G	00	9E	00002	.WORD	Save R2,R3,R4,R5,R6,R7	3729
				56	0000'	CF	9E	00009	MOVAB	LIB\$SIGNAL, R7	
				5E	98	AE	9E	0000E	MOVAB	OUTDSC, R6	
				6E		00	2C	00012	MOVAB	-104(SP), SP	
					F5FC	C6		0001A	MOVCS	#0, (SP), #255, #96, AREA	3778
						7E	D4	0001D	CLRL	-(SP)	3784
					0000V	CF	9F	0001F	PUSHAB	LCPK_GETSERVERS	
				00		02	FB	00023	CALLS	#2, SYS\$CMKRNL	
				53		50	DO	0002A	MOVL	R0, STATUS	
				05		53	E8	0002D	BLBS	STATUS, 1\$	
						53	DD	00030	PUSHL	STATUS	3786
				67		01	FB	00032	CALLS	#1, LIB\$SIGNAL	
				52	FC	A6	DO	00035	1\$: MOVL	AREA_RET_SIZE, R2	3799
						0A	12	00039	BNEQ	2\$	
					00000000G	8F	DD	0003B	PUSHL	#LCP\$ NOSERVERS	3801
				67		01	FB	00041	CALLS	#1, LIB\$SIGNAL	
						04		00044	RET		
				54	F5FC	C6	9E	00045	2\$: MOVAB	AREA, R4	3804
				55	F5E3	C642	9E	0004A	MOVAB	AREA-25[R2], R5	3805
						0092	31	00050	BRW	10\$	3806
				52		54	DO	00053	3\$: MOVL	INDX, PTR	3809
						62	95	00056	TSTB	(PTR)	3814
						07	12	00058	BNEQ	4\$	
				6E	0000'	CF	9E	0005A	MOVAB	UNKNOWN, PARAM_BUF	3816
						10	11	0005F	BRB	5\$	
				6E	60	AE	9E	00061	4\$: MOVAB	NAME_DSC, PARAM_BUF	3819
					62	AE	B4	00065	CLRW	NAME_DSC+2	3820
60	AE					62	9B	00068	MOVZBW	(PTR), NAME_DSC	
64	AE				01	A2	9E	0006C	MOVAB	1(R2), NAME_DSC+4	
				52		11	C0	00071	5\$: ADDL2	#17, PTR	3827
04	AE				01	A2	9A	00074	MOVZBL	1(PTR), PARAM_BUF+4	3832
08	AE				02	A2	9A	00079	MOVZBL	2(PTR), PARAM_BUF+8	3833
0C	AE				03	A2	9A	0007E	MOVZBL	3(PTR), PARAM_BUF+12	3834
10	AE				04	A2	9A	00083	MOVZBL	4(PTR), PARAM_BUF+16	3835
14	AE				05	A2	9A	00088	MOVZBL	5(PTR), PARAM_BUF+20	3836
18	AE				06	A2	9A	0008D	MOVZBL	6(PTR), PARAM_BUF+24	3837
20	AE					62	9A	00092	MOVZBL	(PTR), PARAM_BUF+32	3839
	08				07	A2	E9	00096	BLBC	7(PTR), 6\$	3843
1C	AE				0000'	CF	9E	0009A	MOVAB	ACTIVE, PARAM_BUF+28	3845
						06	11	000A0	BRB	7\$	
1C	AE				0000'	CF	9E	000A2	6\$: MOVAB	INACTIVE, PARAM_BUF+28	3847
					0400	8F	3C	000AB	7\$: MOVZWL	#1024, OUTDSC	3850
04	A6				08	A6	9E	000AD	MOVAB	OUTBUF, OUTDSC+4	
					4040	8F	BB	000B2	PUSHR	#^M<R6,SP>	3864
						56	DD	000B6	PUSHL	R6	
					0000'	CF	9F	000B8	PUSHAB	P.AFY	
						04	FB	000BC	CALLS	#4, SYS\$FAOL	
				00000000G	00						

LATCP
V04-000

LAT Control Program
LCP_SHOWSERVERS Display the LAT Servers

E 14
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 122
(33)

53	50	D0	000C3	MOVL	R0, STATUS	:	
05	53	E8	000C6	BLBS	STATUS, 8\$	:	3866
	53	DD	000C9	PUSHL	STATUS	:	
67	01	FB	000CB	CALLS	#1, LIB\$SIGNAL	:	
	56	DD	000CE	PUSHL	R6	:	3868
00000000G 00	01	FB	000D0	CALLS	#1, LIB\$PUT_OUTPUT	:	
53	50	D0	000D7	MOVL	R0, STATUS	:	
05	53	E8	000DA	BLBS	STATUS, 9\$	:	3870
	53	DD	000DD	PUSHL	STATUS	:	
67	01	FB	000DF	CALLS	#1, LIB\$SIGNAL	:	
54	19	C0	000E2	ADDL2	#25, INDX	:	
55	54	D1	000E5	CMPL	INDX, R5	:	
	03	14	000E8	BGTR	11\$	:	
	FF66	31	000EA	BRW	3\$	:	
	04	000ED	11\$:	RET		:	3878

; Routine Size: 238 bytes, Routine Base: \$CODE\$ + 11E2

```
3775 3879 1 %SBTTL 'LCP_HELP Give HELP'
3776 3880 1 GLOBAL ROUTINE LCP_HELP : NOVALUE =
3777 3881 1
3778 3882 1 ++
3779 3883 1 FUNCTIONAL DESCRIPTION:
3780 3884 1
3781 3885 1 This routine is the LATCP help facility, and will display
3782 3886 1 useful and informative explanations of the LATCP facility.
3783 3887 1
3784 3888 1 CALLING SEQUECE:
3785 3889 1 standard
3786 3890 1
3787 3891 1 INPUT PARAMETERS:
3788 3892 1 none
3789 3893 1
3790 3894 1 IMPLICIT INPUTS:
3791 3895 1 none
3792 3896 1
3793 3897 1 OUTPUTS PARAMETERS:
3794 3898 1 none
3795 3899 1
3796 3900 1 IMPLICIT OUTPUTS:
3797 3901 1
3798 3902 1 The help text will be printed on SYS$OUTPUT.
3799 3903 1
3800 3904 1 ROUTINES CALLED:
3801 3905 1
3802 3906 1 LBR$OUTPUT_HELP
3803 3907 1
3804 3908 1 ROUTINE VALUE:
3805 3909 1 none
3806 3910 1
3807 3911 1 SIGNALS
3808 3912 1 none
3809 3913 1
3810 3914 1 SIDE EFFECTS:
3811 3915 1 none
3812 3916 1
3813 3917 1 --
3814 3918 1
3815 3919 2 BEGIN
3816 3920 2
3817 3921 2 EXTERNAL ROUTINE
3818 3922 2
3819 3923 2 LBR$OUTPUT_HELP : ADDRESSING_MODE(GENERAL);
3820 3924 2
3821 3925 2 BIND
3822 3926 2
3823 3927 2 LIBRARY_NAME = DESCRIPTOR ('LATCP'), ! HELP text library
3824 3928 2 KEYWORDS_DSC = DESCRIPTOR ('KEYWORDS'); ! Keywords desc
3825 3929 2
3826 3930 2 LOCAL
3827 3931 2
3828 3932 2 SUBCMD_DSC : BLOCK [DSC$K_Z_BLN, BYTE], ! Rest of line
3829 3933 2 STATUS;
3830 3934 2
3831 3935 2
```



```
3832 3936 2 |
3833 3937 2 |
3834 3938 2 |
3835 3939 2 |
3836 3940 2 |
3837 3941 2 |
3838 3942 2 |
3839 3943 2 |
3840 3944 2 |
3841 3945 2 |
3842 3946 2 |
3843 3947 2 |
3844 3948 2 |
3845 3949 2 |
3846 3950 2 |
3847 3951 1 |

Get help keywords and uppercase them

CH$FILL (0, DSC$C_2 BLN, SUBCMD DSC); ! Zero fill desc
SUBCMD DSC [DSC$B-CLASS] = DSC$R CLASS D; ! Fill in desc class
CL$GET VALUE (KEYWORDS DSC, SUBCMD DSC); ! Get the KEYWORDS
IF .SUBCMD_DSC [DSC$W_LENGTH] NEQ 0-
THEN
    STR$UPCASE (SUBCMD_DSC, SUBCMD_DSC);

Call LBR$OUTPUT_HELP to do all the real work.

LBR$OUTPUT_HELP (LIB$PUT_OUTPUT, 0, SUBCMD DSC,
    LIBRARY_NAME, 0, LIB$GET_INPUT);

END;
```

```

.PSECT $SPLITS,NOWRT,NOEXE,2

50 43 54 41 4C 00FC0 P.AGB: .ASCII \LATCP\
00FC5 .BLKB 3
00000005 00FC8 P.AGA: .LONG 5
00000000 00FCC .ADDRESS P.AGB
53 44 52 4F 57 59 45 4B 00FD0 P.AGD: .ASCII \KEYWORDS\
00000008 00FDB P.AGC: .LONG 8
00000000 00FDC .ADDRESS P.AGD
```

```
LIBRARY_NAME= P.AGA
KEYWORDS_DSC= P.AGC
-EXTRN LBR$OUTPUT_HELP
```

```
.PSECT $CODE$,NOWRT,2
```

```
08      00      5E      08 003C 00000
      6E      00 00 00 00002
      03 AE      00 00 2C 00005
      00000000G 00 00 00 0000A
      00000000G 00 00 00 0000B
      00000000G 00 00 00 0000F
      00000000G 00 00 00 00011
      00000000G 00 00 00 00015
      00000000G 00 00 00 0001C
      00000000G 00 00 00 0001E
      00000000G 00 00 00 00020
      00000000G 00 00 00 00022
      00000000G 00 00 00 00025
      00000000G 00 00 00 0002C 1$:
      00000000G 00 00 00 00032
      00000000G 00 00 00 00034
      00000000G 00 00 00 00038
      00000000G 00 00 00 0003B
      00000000G 00 00 00 0003D
      00000000G 00 00 00 00043
      00000000G 00 00 00 0004A

.ENTRY LCP_HELP, Save R2,R3,R4,R5
SUBL2 #8, SP
MOVCS #0, (SP), #0, #8, SUBCMD_DSC

MOVB #2, SUBCMD_DSC+3
PUSHL SP
PUSHAB KEYWORDS_DSC
CALLS #2, CL$GET_VALUE
TSTW SUBCMD_DSC
BEQL 1$
PUSHL SP
PUSHAB SUBCMD_DSC
CALLS #2, STR$UPCASE
PUSHAB LIB$GET_INPUT
CLRL -(SP)
PUSHAB LIBRARY_NAME
PUSHAB SUBCMD_DSC
CLRL -(SP)
PUSHAB LIB$PUT_OUTPUT
CALLS #6, LBR$OUTPUT_HELP
RET
```

```
3880
3939
3940
3941
3942
3944
3949
3951
```

LATCP
V04-000

LAT Control Program
LCP_HELP Give HELP

H 14
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1 Page 125
(34)

; Routine Size: 75 bytes, Routine Base: \$CODE\$ + 12D0

; 3848 3952 1

LATCP
V04-000

LAT Control Program
LCP_FAOUTPUT Convert and write a string

I 14
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1 Page 126
(35)

```

3850 3953 1 %SBTTL 'LCP_FAOUTPUT Convert and write a string'
3851 3954 1 ROUTINE LCP_FAOUTPUT (FAOSTR, DATA) : NOVALUE =
3852 3955 1
3853 3956 1
3854 3957 1 **
3855 3958 1 FUNCTIONAL DESCRIPTION:
3856 3959 1 Convert 32 bytes to a suitable parameter list for fao. The fao
3857 3960 1 parameter string and the address of the data are passed.
3858 3961 1
3859 3962 1 FORMAL PARAMETERS:
3860 3963 1
3861 3964 1 FAOSTR Address of desc of fao control string
3862 3965 1 DATA Address of 8 bytes of data
3863 3966 1
3864 3967 1 IMPLICIT INPUTS:
3865 3968 1
3866 3969 1 NONE
3867 3970 1
3868 3971 1 IMPLICIT OUTPUTS:
3869 3972 1
3870 3973 1 NONE
3871 3974 1
3872 3975 1 ROUTINE VALUE:
3873 3976 1 COMPLETION CODES:
3874 3977 1
3875 3978 1 NONE
3876 3979 1
3877 3980 1 SIDE EFFECTS:
3878 3981 1
3879 3982 1 NONE
3880 3983 1
3881 3984 1 --
3882 3985 1
3883 3986 2 BEGIN
3884 3987 2
3885 3988 2 MAP
3886 3989 2 DATA : REF VECTOR [, BYTE]
3887 3990 2 ;
3888 3991 2
3889 3992 2 LOCAL
3890 3993 2 STATUS,
3891 3994 2 EXPLODE : VECTOR [32, LONG]
3892 3995 2 ;
3893 3996 2
3894 3997 2
3895 3998 2 | Set the output buffer descriptor
3896 3999 2
3897 4000 2 $SETDSC (OUTDSC, OUTBUFSIZE, OUTBUF);
3898 4001 2
3899 4002 2
3900 4003 2 | Explode the data to longwords eight bytes of a time.
3901 4004 2
3902 4005 2 IF .DATA NEQ 0
3903 4006 2 THEN
3904 4007 2 BEGIN
3905 4008 2 EXPLODE [ 0] = DATA [0]; ! Get address of time
3906 4009 2 INCR IDX FROM 8 TO 31
```

पुस्तकें

		001C 00000 LCP_FA00		OUTPUT:		
	54	00000000G	00	9E 00002	.WORD	3954
	53	0000	CF	9E 00009	MOVAB	LIB\$SIGNAL, R4
	5E	80	AE	9E 0000E	MOVAB	OUTDSC, R3
	63	0400	8F	3C 00012	MOVAB	-128(SP), SP
04	A3	08	A3	9E 00017	MOVZWL	#1024, OUTDSC
	51	08	AC	D0 0001C	MOVAB	OUTBUF, OUTDSC+4
			10	13 00020	MOVL	DATA, R1
	6E		51	D0 00022	BEQL	2\$
	50		08	D0 00025	MOVL	R1, EXPLODE
	E4	AE40	6041	9A 00028	MOVL	#8, IDX
F6	50		1F	F3 0002E	MOVZBL	(IDX)[R1], EXPLODE-28[IDX]
		4008	8F	BB 00032	AOBLEQ	#31, IDX, 1\$
			53	DD 00036	PUSHR	#^M<R3, SP>
		04	AC	DD 00038	PUSHL	R3
00000000G	00		04	FB 0003B	PUSHL	FAOSTR
	52		50	D0 00042	CALLS	#4, SYSS\$FAOL
	05		52	E8 00045	MOVL	R0, STATUS
			52	DD 00048	BLBS	STATUS, 3\$
	64		01	FB 0004A	PUSHL	STATUS
			53	DD 0004D	CALLS	#1, LIB\$SIGNAL
00000000G	00		01	FB 0004F	PUSHL	R3
	52		50	D0 00056	CALLS	#1, LIB\$PUT_OUTPUT
	05		52	E8 00059	MOVL	R0, STATUS
			52	DD 0005C	BLBS	STATUS, 4\$
	64		01	FB 0005E	PUSHL	STATUS
					CALLS	#1, LIB\$SIGNAL

LATCP
V04-000

LAT Control Program
LCP_FA00OUTPUT Convert and write a string

K 14
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMMASTER:[LAT.SRC]LATCP.B32;1

Page 128
(35)

04 00061 4\$: RET

; 4034

; Routine Size: 98 bytes, Routine Base: \$CODE\$ + 131B

**

LATCP
V04-000

LAT Control Program
LCPK_GETCOUNTERS Obtain the counterblock

L 14
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1 Page 129
(36)

```
: 3933 4035 1 %SBTTL 'LCPK_GETCOUNTERS Obtain the counterblock'
: 3934 4036 1 ROUTINE LCPK_GETCOUNTERS =
: 3935 4037 1
: 3936 4038 1 ++
: 3937 4039 1 FUNCTIONAL DESCRIPTION:
: 3938 4040 1
: 3939 4041 1 Copy the counters from the driver, to an area here for conversion.
: 3940 4042 1
: 3941 4043 1 FORMAL PARAMETERS:
: 3942 4044 1
: 3943 4045 1 NONE
: 3944 4046 1
: 3945 4047 1 IMPLICIT INPUTS:
: 3946 4048 1
: 3947 4049 1 counter block in driver
: 3948 4050 1
: 3949 4051 1 IMPLICIT OUTPUTS:
: 3950 4052 1
: 3951 4053 1 counter block here
: 3952 4054 1
: 3953 4055 1 ROUTINE VALUE:
: 3954 4056 1 COMPLETION CODES:
: 3955 4057 1
: 3956 4058 1 success or failure if no driver
: 3957 4059 1
: 3958 4060 1 SIDE EFFECTS:
: 3959 4061 1
: 3960 4062 1 NONE
: 3961 4063 1
: 3962 4064 1 --
: 3963 4065 1
: 3964 4066 2 BEGIN
: 3965 4067 2
: 3966 4068 2 CH$MOVE (GHBSK_LENGTH, .LCP$L_AREA, AREA);
: 3967 4069 2 SUCCESS
: 3968 4070 2
: 3969 4071 1 END;
```

003C 00000 LCPK_GETCOUNTERS:

```
0000' CF 0000' DF 0060 8F 28 00002 .WORD Save R2,R3,R4,R5
01 D0 0000C MOVCL #96, @LCP$L_AREA, AREA
04 0000F MOVL #1, R0
RET
```

```
: 4036
: 4068
: 4071
:
```

; Routine Size: 16 bytes, Routine Base: \$CODE\$ + 137D


```

3971 4072 1 %SBTTL 'LCPK_GETSERVCTRS Obtain the SERVER counters'
3972 4073 1 ROUTINE LCPK_GETSERVCTRS =
3973 4074 1
3974 4075 1 ++
3975 4076 1 FUNCTIONAL DESCRIPTION:
3976 4077 1
3977 4078 1     Copy the counters from the driver, to an area here for conversion.
3978 4079 1
3979 4080 1 FORMAL PARAMETERS:
3980 4081 1
3981 4082 1     NONE
3982 4083 1
3983 4084 1 IMPLICIT INPUTS:
3984 4085 1
3985 4086 1     counter block in driver
3986 4087 1
3987 4088 1 IMPLICIT OUTPUTS:
3988 4089 1
3989 4090 1     counter block here
3990 4091 1
3991 4092 1 ROUTINE VALUE:
3992 4093 1 COMPLETION CODES:
3993 4094 1
3994 4095 1     success or failure if no driver
3995 4096 1
3996 4097 1 SIDE EFFECTS:
3997 4098 1
3998 4099 1     NONE
3999 4100 1
4000 4101 1 --
4001 4102 1
4002 4103 2 BEGIN
4003 4104 2
4004 4105 2 LOCAL
4005 4106 2     CSB_LIST : REF VECTOR [, LONG],
4006 4107 2     CSB_PTR : REF VECTOR [, BYTE],
4007 4108 2     CSB_HEADER,
4008 4109 2     PTR
4009 4110 2 ;
4010 4111 2
4011 4112 2 |
4012 4113 2 |     Point to output buffer
4013 4114 2 |
4014 4115 2 | PTR = AREA;
4015 4116 2 |
4016 4117 2 |
4017 4118 2 |     Get pointer to CSB table
4018 4119 2 |
4019 4120 2 | CSB_LIST = .LCP$L_AREA [GHB$L_CSBLST];
4020 4121 2 |
4021 4122 2 |
4022 4123 2 |     Get the counters from each active CSB
4023 4124 2 |
4024 4125 2 | IF .CSB_LIST NEQ 0
4025 4126 2 | THEN
4026 4127 2 |     INCR INDX FROM 0 TO GHB$K_ACT_CSBS - 1
4027 4128 2 | DO
```

```

: 4028      4129      3      BEGIN
: 4029      4130      3      CSB_PTR = .CSB_LIST [.INDX];
: 4030      4131      3      IF .CSB_PTR NEQ 0
: 4031      4132      3      THEN
: 4032      4133      3      PTR = CH$MOVE (LCB_C_LENGTH + GHBSK_NAMELEN + 1,
: 4033      4134      3      CSB_PTR [GHBSK_CSBCTR], .PTR)
: 4034      4135      3      END
: 4035      4136      2      ;
: 4036      4137      2      :
: 4037      4138      2      : Get pointer to inactive CSB queue, go through queue in reverse order.
: 4038      4139      2      CSB_HEADER = LCP$L_AREA [GHBSK_OLD_CSBS];
: 4039      4140      2
: 4040      4141      2      :
: 4041      4142      2      : Get the counters from each inactive CSB
: 4042      4143      2
: 4043      4144      2      IF ..CSB_HEADER NEQ 0
: 4044      4145      2      THEN
: 4045      4146      2      BEGIN
: 4046      4147      2      CSB_PTR = .(.CSB_HEADER+4);
: 4047      4148      3      WHILE .CSB_PTR NEQ .CSB_HEADER
: 4048      4149      3      DO
: 4049      4150      3      BEGIN
: 4050      4151      3      PTR = CH$MOVE (LCB_C_LENGTH + GHBSK_NAMELEN + 1,
: 4051      4152      4      CSB_PTR [GHBSK_CSBCTR], .PTR);
: 4052      4153      4      CSB_PTR = .(.CSB_PTR+4)
: 4053      4154      4      END
: 4054      4155      5
: 4055      4156      4
: 4056      4157      3      :
: 4057      4158      3      END
: 4058      4159      2      :
: 4059      4160      2      : Calculate size of area filled in
: 4060      4161      2      AREA_RET_SIZE = .PTR - AREA;
: 4061      4162      2
: 4062      4163      2      SUCCESS
: 4063      4164      2
: 4064      4165      2
: 4065      4166      2
: 4066      4167      1      END;
```

```

                                03FC 0000 LCPK_GETSERVCTRS:
                                .WORD      Save R2,R3,R4,R5,R6,R7,R8,R9
53      0000'   CF   9E 00002      MOVAB      AREA, PTR      : 4073
56      0000'   CF   D0 00007      MOVL       LCP$L_AREA, R6      : 4115
59      20      A6   D0 0000C      MOVL       32(R6), CSB_LIST      : 4120
                                11   13 00010      BEQL       3$      : 4125
                                58   D4 00012      CLRL       INDX      : 4127
57      6948   D0 00014 1$:      MOVL       (CSB_LIST)[INDX], CSB_PTR      : 4130
                                05   13 00018      BEQL       2$      : 4131
63      0C      A7   21 28 0001A      MOV(C3    #33, 12(CSB_PTR), (PTR)      : 4134
F1      58      1F   F3 0001F 2$:      AOBLEQ    #31, INDX, T$      : 4131
56      24      C0 00023 3$:      ADDL2      #36, CSB_HEADER      : 4141
                                66   D5 00026      TSTL       (CSB_HEADER)      : 4146
```


LATCP
V04-000

LAT Control Program
LCPK_GETSERVCTRS Obtain the SERVER counters

B 15
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 132
(37)

				14	13	00028	BEQL	5\$	:		
		57	04	A6	D0	0002A	MOVL	4(CSB_HEADER), CSB_PTR	:	4149	
		56		57	D1	0002E	4\$:	CPL	CSB_PTR, CSB_HEADER	:	4150
				0B	13	00031	BEQL	5\$	:		
63	0C	A7		21	28	00033	MOVC3	#33, 12(CSB_PTR), (PTR)	:	4154	
		57	04	A7	D0	00038	MOVL	4(CSB_PTR), CSB_PTR	:	4155	
				F0	11	0003C	BRB	4\$	:		
		50	0000'	CF	9E	0003E	5\$:	MOVAB	AREA, R0	:	4163
0000'	CF	53		50	C3	00043	SUBL3	R0, PTR, AREA_RET_SIZE	:		
		50		01	D0	00049	MOVL	#1, R0	:	4167	
				04	0004C		RET		:		

; Routine Size: 77 bytes, Routine Base: \$CODE\$ + 138D


```

: 4068 4168 1 %SBTTL 'LCPK_GETSERVERS Obtain the SERVER information'
: 4069 4169 1 ROUTINE LCPK_GETSERVERS =
: 4070 4170 1
: 4071 4171 1 !++
: 4072 4172 1 FUNCTIONAL DESCRIPTION:
: 4073 4173 1
: 4074 4174 1 Copy the CSB server info from the driver, to an area here for conversion.
: 4075 4175 1
: 4076 4176 1 FORMAL PARAMETERS:
: 4077 4177 1
: 4078 4178 1 NONE
: 4079 4179 1
: 4080 4180 1 IMPLICIT INPUTS:
: 4081 4181 1
: 4082 4182 1 CSB server block in driver
: 4083 4183 1
: 4084 4184 1 IMPLICIT OUTPUTS:
: 4085 4185 1
: 4086 4186 1 CSB server block info here
: 4087 4187 1
: 4088 4188 1 ROUTINE VALUE:
: 4089 4189 1 COMPLETION CODES:
: 4090 4190 1
: 4091 4191 1 success or failure if no driver
: 4092 4192 1
: 4093 4193 1 SIDE EFFECTS:
: 4094 4194 1
: 4095 4195 1 NONE
: 4096 4196 1
: 4097 4197 1 --
: 4098 4198 1
: 4099 4199 2 BEGIN
: 4100 4200 2
: 4101 4201 2 LOCAL
: 4102 4202 2 CSB_LIST : REF VECTOR [, LONG],
: 4103 4203 2 CSB_PTR : REF VECTOR [, BYTE],
: 4104 4204 2 CSB_HEADER,
: 4105 4205 2 PTR,
: 4106 4206 2 ACTIVE_V,
: 4107 4207 2 INACTIVE_V
: 4108 4208 2 ;
: 4109 4209 2
: 4110 4210 2 ACTIVE_V = 1; INACTIVE_V = 0;
: 4111 4211 2
: 4112 4212 2 Point to output buffer
: 4113 4213 2
: 4114 4214 2 PTR = AREA;
: 4115 4215 2
: 4116 4216 2
: 4117 4217 2 Get pointer to CSB table
: 4118 4218 2
: 4119 4219 2 CSB_LIST = .LCP$L_AREA [GHB$L_CSBLST];
: 4120 4220 2
: 4121 4221 2
: 4122 4222 2 Get the info from each active CSB
: 4123 4223 2
: 4124 4224 2 IF .CSB_LIST NEQ 0
```



```

: 4125      4225 2      THEN
: 4126      4226 2      INCR INDX FROM 0 TO GHBSK_ACT_CSBS - 1
: 4127      4227 2      DO
: 4128      4228 3      BEGIN
: 4129      4229 3      CSB_PTR = .CSB_LIST [.INDX];
: 4130      4230 3      IF .CSB_PTR NEQ 0
: 4131      4231 3      THEN
: 4132      4232 4      BEGIN
: 4133      4233 4      PTR = CH$MOVE (GHBSK_NAMELEN + 8,
: 4134      4234 4      CSB_PTR [GHBSK_CSBCTR + LCB_C_LENGTH], .PTR);
: 4135      4235 4      PTR = CH$MOVE (1, ACTIVE_V, .PTR)
: 4136      4236 4      END
: 4137      4237 3      ;
: 4138      4238 3      END
: 4139      4239 2      ;
: 4140      4240 2      ;
: 4141      4241 2      ;
: 4142      4242 2      ; Get pointer to inactive CSB queue, go through queue in reverse order.
: 4143      4243 2      CSB_HEADER = LCP$L_AREA [GHBSK_OLD_CSBS];
: 4144      4244 2      ;
: 4145      4245 2      ;
: 4146      4246 2      ; Get the info from each inactive CSB
: 4147      4247 2      ;
: 4148      4248 2      ;
: 4149      4249 2      IF ..CSB_HEADER NEQ 0
: 4150      4250 2      THEN
: 4151      4251 3      BEGIN
: 4152      4252 3      CSB_PTR = .(.CSB_HEADER+4);
: 4153      4253 3      WHILE .CSB_PTR NEQ .CSB_HEADER
: 4154      4254 3      DO
: 4155      4255 4      BEGIN
: 4156      4256 4      PTR = CH$MOVE (GHBSK_NAMELEN + 8,
: 4157      4257 4      CSB_PTR [GHBSK_CSBCTR + LCB_C_LENGTH], .PTR);
: 4158      4258 4      PTR = CH$MOVE (1, INACTIVE_V, .PTR);
: 4159      4259 5      CSB_PTR = .(.CSB_PTR+4)
: 4160      4260 4      END
: 4161      4261 3      ;
: 4162      4262 3      END
: 4163      4263 2      ;
: 4164      4264 2      ;
: 4165      4265 2      ;
: 4166      4266 2      ; Calculate size of area filled in
: 4167      4267 2      ;
: 4168      4268 2      AREA_RET_SIZE = .PTR - AREA;
: 4169      4269 2      ;
: 4170      4270 2      SUCCESS
: 4171      4271 2      ;
: 4172      4272 1      END;
```

OFFC 00000 LCPK_GETSERVERS:

5A	0000'	01	7D 00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	: 4169
53		CF	9E 00005	MOVQ	#1, ACTIVE_V	: 4210
				MOVAB	AREA, PTR	: 4214

LATCP
V04-000

LAT Control Program
LCPK_GETSERVERS Obtain the SERVER information

E 15
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 135
(38)

		56	0000'	CF	D0	0000A	MOVL	LCPSL_AREA, R6	:	4219	
		59	20	A6	D0	0000F	MOVL	32(R6), CSB_LIST	:		
				14	13	00013	BEQL	3\$	:	4224	
				58	D4	00015	CLRL	INDX	:	4226	
		57	6948	D0	00017	1\$:	MOVL	(CSB_LIST)[INDX], CSB_PTR	:	4229	
				08	13	0001B	BEQL	2\$	:	4230	
63	1C	A7		18	28	0001D	MOVC3	#24, 28(CSB_PTR), (PTR)	:	4234	
		83		5A	90	00022	MOVB	ACTIVE V, (PTR)+	:	4235	
EE		58		1F	F3	00025	AOBLEQ	#31, INDX, 1\$	:	4226	
		56		24	C0	00029	ADDL2	#36, CSB_HEADER	:	4244	
				66	D5	0002C	TSTL	(CSB_HEADER)	:	4249	
				17	13	0002E	BEQL	5\$	:		
		57	04	A6	D0	00030	MOVL	4(CSB_HEADER), CSB_PTR	:	4252	
		56		57	D1	00034	4\$:	CML	CSB_PTR, CSB_HEADER	:	4253
				0E	13	00037	BEQL	5\$	:		
63	1C	A7		18	28	00039	MOVC3	#24, 28(CSB_PTR), (PTR)	:	4257	
		83		5B	90	0003E	MOVB	INACTIVE V, (PTR)+	:	4258	
		57	04	A7	D0	00041	MOVL	4(CSB_PTR), CSB_PTR	:	4259	
				ED	11	00045	BRB	4\$	:		
		50	0000'	CF	9E	00047	5\$:	MOVAB	AREA, R0	:	4268
		53		50	C3	0004C	SUBL3	R0, PTR, AREA_RET_SIZE	:		
		50		01	D0	00052	MOVL	#1, R0	:	4272	
				04	00	00055	RET		:		

; Routine Size: 86 bytes, Routine Base: \$CODE\$ + 13DA


```
4174 4273 1 %SBTTL 'LCPK_GETHISTORY Obtain the history buffer'
4175 4274 1 ROUTINE LCPK_GETHISTORY =
4176 4275 1
4177 4276 1 ++
4178 4277 1 FUNCTIONAL DESCRIPTION:
4179 4278 1
4180 4279 1 Copy the history buffer from the driver, to an area here for conversion.
4181 4280 1
4182 4281 1 FORMAL PARAMETERS:
4183 4282 1
4184 4283 1 NONE
4185 4284 1
4186 4285 1 IMPLICIT INPUTS:
4187 4286 1
4188 4287 1 counter block in driver
4189 4288 1
4190 4289 1 IMPLICIT OUTPUTS:
4191 4290 1
4192 4291 1 counter block here
4193 4292 1
4194 4293 1 ROUTINE VALUE:
4195 4294 1 COMPLETION CODES:
4196 4295 1
4197 4296 1 NONE
4198 4297 1
4199 4298 1 SIDE EFFECTS:
4200 4299 1
4201 4300 1 NONE
4202 4301 1
4203 4302 1 --
4204 4303 1
4205 4304 2 BEGIN
4206 4305 2
4207 4306 2 LOCAL
4208 4307 2 HIST_BUF : REF BLOCK [,BYTE]
4209 4308 2 ;
4210 4309 2
4211 4310 2
4212 4311 2 If no history buffer, return error
4213 4312 2
4214 4313 2 IF .LCP$ AREA [GHBSL_HISTORY] EQL 0
4215 4314 2 THEN
4216 4315 2 RETURN LCP$_NOHISTORY
4217 4316 2 ;
4218 4317 2
4219 4318 2
4220 4319 2 Copy the history buffer
4221 4320 2
4222 4321 2 HIST_BUF = .LCP$ AREA [GHBSL_HISTORY];
4223 4322 2 CH$MOVE (GHBSL_HISTSIZE,
4224 4323 2 HIST_BUF [RBF_Z_DATA], AREA);
4225 4324 2
4226 4325 2 SUCCESS
4227 4326 2
4228 4327 1 END;
```

LATCP
V04-000

LAT Control Program
LCPK_GETHISTORY Obtain the history buffer

G 15
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 137
(39)

```

                                003C 00000 LCPK_GETHISTORY:
                                .WORD      Save R2,R3,R4,R5
50      0000'  CF  D0 00002      MOVL     LCP$L_AREA, R0      : 4274
      08      A0  D5 00007      TSTL     8(R0)                : 4313
      08      12 0000A      BNEQ     1$
50 00000000G  8F  D0 0000C      MOVL     #LCP$_NOHISTORY, R0    : 4315
      04 00013      RET
0000'  CF      50      08      A0  D0 00014 1$:      MOVL     8(R0), HIST_BUF      : 4321
      A0      CA00  8F  28 00018      MOVC3   #2560, 12(HIST_BUF), AREA : 4323
      50      01  D0 00021      MOVL     #1, R0                : 4327
      04 00024      RET

```

; Routine Size: 37 bytes, Routine Base: \$CODE\$ + 1430


```
4230 4328 1 %SBTTL 'LCPK_GETCOMMON Kernel mode routine to get common buffer area'
4231 4329 1 ROUTINE LCPK_GETCOMMON =
4232 4330 1
4233 4331 1 ++
4234 4332 1 FUNCTIONAL DESCRIPTION:
4235 4333 1
4236 4334 1 Kernel mode routine to get the information from the common area in
4237 4335 1 driver.
4238 4336 1
4239 4337 1 FORMAL PARAMETERS:
4240 4338 1
4241 4339 1 NONE
4242 4340 1
4243 4341 1 IMPLICIT INPUTS:
4244 4342 1
4245 4343 1 results of the qualifier parse
4246 4344 1
4247 4345 1 IMPLICIT OUTPUTS:
4248 4346 1
4249 4347 1 NONE
4250 4348 1
4251 4349 1 ROUTINE VALUE:
4252 4350 1 COMPLETION CODES:
4253 4351 1
4254 4352 1 NONE
4255 4353 1
4256 4354 1 SIDE EFFECTS:
4257 4355 1
4258 4356 1 qualifier results set for multicast
4259 4357 1
4260 4358 1 --
4261 4359 1
4262 4360 2 BEGIN
4263 4361 2
4264 4362 2 LOCAL
4265 4363 2 LT : REF BLOCK [,BYTE],
4266 4364 2 PTR,
4267 4365 2 STR : REF BLOCK [,BYTE]
4268 4366 2 ;
4269 4367 2
4270 4368 2 !
4271 4369 2 ! Setup the output descriptor
4272 4370 2
4273 4371 2 $SETDSC (OUTDSC, OUTBUFSIZE, OUTBUF);
4274 4372 2
4275 4373 2 !
4276 4374 2 ! Copy the common area
4277 4375 2
4278 4376 2 PTR = CH$MOVE (GHB$$_COMM_AREA, .LCP$L_AREA, AREA);
4279 4377 2
4280 4378 2 !
4281 4379 2 ! Copy the node information to the output buffer area
4282 4380 2
4283 4381 2 LT = .LCP$L_AREA;
4284 4382 2 STR = .LT [GHB$L_NODE];
4285 4383 2 PTR = .OUTDSC [DSC$A_POINTER];
4286 4384 2 PTR = CH$MOVE (.STR [-2, 0, 16, 0], .LT [GHB$L_NODE], .PTR);
```

LATCP
V04-000

LAT Control Program
LCPK_GETCOMMON Kernel mode routine to get comm

I 15
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1

Page 139
(40)

: 4287
: 4288
: 4289
: 4290
4385 2
4386 2
4387 2
4388 1
SUCCESS
END;

007C 00000 LCPK_GETCOMMON:

			56	0000'	CF	9E	00002	.WORD	Save R2,R3,R4,R5,R6	: 4329
		FC	A6	0400	8F	3C	00007	MOVAB	OUTDSC+4, R6	: 4371
			66	04	A6	9E	0000D	MOVZWL	#1024, OUTDSC	: 4376
F5F8	C6	F5F4	D6	0060	8F	28	00011	MOVAB	OUTBUF, OUTDSC+4	: 4381
			50	F5F4	C6	D0	0001B	MOVAB	#96, @LCP\$ AREA, AREA	: 4382
			51	1C	A0	D0	00020	MOVAB	LCP\$ AREA, LT	: 4383
			53		66	D0	00024	MOVAB	28(LT), STR	: 4384
	63	1C	B0	FE	A1	28	00027	MOVAB	OUTDSC+4, PTR	: 4388
			50		01	D0	0002D	MOVAB	-2(STR), @28(LT), (PTR)	: 4388
					04	00030	RET		#1, R0	:

; Routine Size: 49 bytes, Routine Base: \$CODE\$ + 1455

LATCP
V04-000

LAT Control Program
LCPK_GETCOMMON Kernel mode routine to get comm

J 15
16-Sep-1984 01:49:16
14-Sep-1984 12:37:25

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[LAT.SRC]LATCP.B32;1 Page 140
(41)

: 4292
: 4293
4389 1 END
4390 0 ELUDOM
!End of module

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
\$SPLITS	4064	NOVEC,NOWRT, RD ,NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
\$OWNS	5968	NOVEC, WRT, RD ,NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
\$CODES	5254	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	48	0	1000	00:01.8
\$255\$DUA28:[SHRLIB]NMALIBRY.L32;1	887	35	3	47	00:01.4

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:LATCP/OBJ=OBJ\$:LATCP MSRC\$:LATCP/UPDATE=(ENH\$:LATCP)

: Size: 5254 code + 10032 data bytes
: Run Time: 01:52.9
: Elapsed Time: 03:55.0
: Lines/CPU Min: 2332
: Lexemes/CPU-Min: 20161
: Memory Used: 304 pages
: Compilation Complete

0196

AH-BT13A-SE
 VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY